NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN APPLICATION OF KALMAN FILTERING
AND SMOOTHING TO TORPEDO TRACKING

by

Mustafa Işık

October 1982

Thesis Advisor: H. A. Titus

Approved for public release; distribution unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AN APPLICATION OF KALMAN FILTERING AND SMOOTHING TO TORPEDO TRACKING		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; October 1982
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Mustafa Işık		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE October 1982
		13. NUMBER OF PAGES 115
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Torpedo Tracking
Kalman Filtering
Optimal Smoothing

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A sequential Extended Kalman Filter and Smoothing routine was developed to provide real time estimates of torpedo position and depth on the three dimensional underwater tracking range at the Naval Torpedo Station, Keyport, Washington. Inputs to the routine were acoustic pulse transit times from the target to receiving array elements which are non-linear functions of the position coordinates. These inputs were linearized and

DD FORM 1473, 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

4. TITLE (and Subtitle)

AN APPLICATION OF [illegible]

[illegible]

7. AUTHOR(s)

[illegible]

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Naval Postgraduate School

Monterey, California 93940

11. CONTROLLING OFFICE NAME AND ADDRESS

Naval Postgraduate School

Monterey, California 93940

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; [illegible]

19. KEY WORDS

[illegible]

20. ABSTRACT

[illegible]

DD FORM 1473

20. (Continued)

the filter gains and filtered estimates calculated on-line. By using a smoothing subroutine, all past filtered estimates were smoothed. Tests were conducted using simulated torpedo trajectories that traversed multiple hydrophone arrays. It was found that filter performance was dependent on system noise and the distance the torpedo was from the hydrophone array and the smoothed estimates of states were better than or equal to the filtered estimates.

Approved for public release; distribution unlimited

An Application of Kalman Filtering and Smoothing to Torpedo Tracking

by

Mustafa Işık
Lieutenant Junior Grade, Turkish Navy
B.S.E.E., Naval Postgraduate School, 1982

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
October 1982

ABSTRACT

A sequential Extended Kalman Filter and Smoothing routine was developed to provide real time estimates of torpedo position and depth on the three dimensional underwater tracking range at the Naval Torpedo Station, Keyport, Washington. Inputs to the routine were acoustic pulse transit times from the target to receiving array elements which are non-linear functions of the position coordinates. These inputs were linearized and the filter gains and filtered estimates calculated on-line. By using a smoothing subroutine, all past filtered estimates were smoothed. Tests were conducted using simulated torpedo trajectories that traversed multiple hydrophone arrays. It was found that filter performance was dependent on system noise and the distance the torpedo was from the hydrophone array and the smoothed estimates of states were better than or equal to the filtered estimates.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

The author is deeply indebted to Professor Hal Titus of the Naval Postgraduate School, Monterey, California for his counsel and professional guidance during this project.

I. INTRODUCTION

The NUWES at Keyport, Washington currently operates two three-dimensional (3-D) underwater tracking range utilizing a sonar transmitter installed in the torpedo to be tracked. The transmitter is synchronized with a master clock. Timed acoustic pulses are received by bottom mounted hydrophone arrays and then relayed via cable to a computer at the observation site. The computer calculates the positional coordinates of the torpedo and plots its trajectory through the water.

The measured data, which consists of the elapsed time from transmission of a pulse until its receipt at the hydrophone array, is corrupted with noise due to the combined effects of environmental factors and measurement instruments.

These noisy tracks are later analyzed, and measurements judged most inaccurate on the basis of total track statistics are removed in order to obtain a smooth representation of the track.

An opportunity exists for expanding the capability of the system by applying a real time Kalman Filter and post test Smoothing routine which can take as an input the transit times of the acoustic pulses, and produce the best

estimate of the position of the tracked object at a particular time. Previous research in this area [3] and [4], revealed that a Kalman filter utilizing a sequential estimation approach was desirable.

The intention is to develop and test a sequential Kalman filter and smoothing algorithm that can be interfaced with the current underwater range system.

II. DESCRIPTION OF RANGE TRACKING GEOMETRY

The hydrophone array, consisting of four independent elements, defines an orthogonal coordinate system in which transit time measurements are made. As shown in Figure 1, four hydrophones X, Y, Z, and C are on four adjacent vertices separated by a distance d, along the edge of the cube. The origin of the array coordinates is at the center of the cube with the orthogonal coordinates parallel to its edge. Positional information is computed from the transit times of a periodic synchronous acoustic signal travelling from the torpedo to the four hydrophones on the array. The range measures the tracked torpedo's position every 1.31 seconds to an accuracy that is typically within 3 to 30 feet. A more detailed description of the range tracking capability is described in [2].

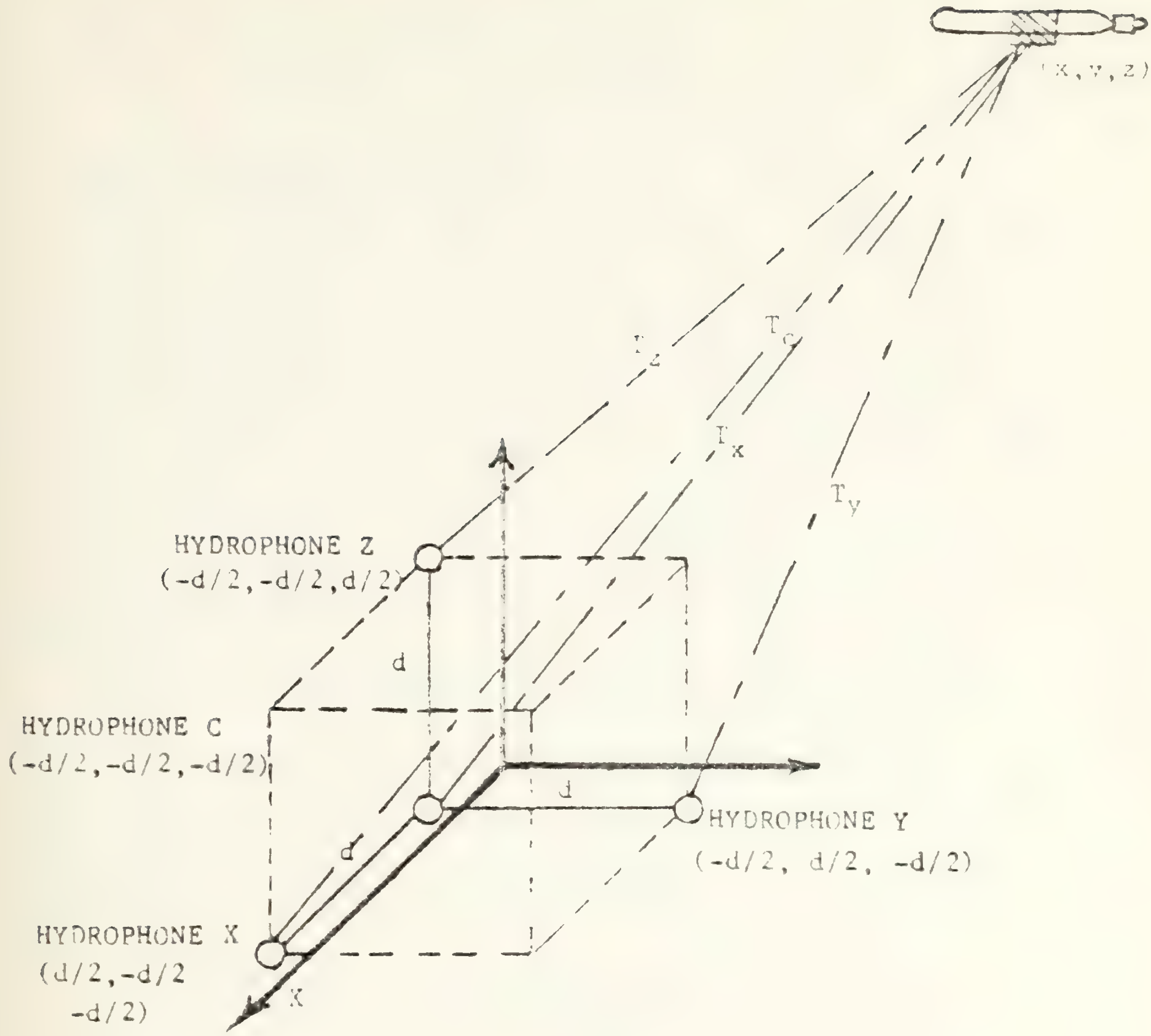


Figure 1. Geometry of a Tracking Array

III. THEORY

A. THE EXTENDED KALMAN FILTER

Since the transit times were readily available and are nonlinear functions of position, these equations can be linearized and Kalman filter theory applied using the extended Kalman filter. This procedure produces a real-time system, filtering on the transit times T_c, T_x, T_y and T_z, without the necessity of converting these times to positions.

For the three-dimensional location problem three position states (x, y, z) and two velocity states (v_x, v_y) specify target motion. The discrete linear and nonlinear observation equations are given by

$$\underline{x}(k + 1) = \Phi \cdot \underline{x}(k) + \Gamma \cdot \underline{w}(k) \tag{3.1}$$

and

$$z(k) = \underline{h}(x(k), k) + \underline{v}(k) \tag{3.2}$$

In these equations Φ and Γ are constant matrices and h is a nonlinear function of the state variable $\underline{x}$. $\underline{w}(k)$ is plant excitation noise and $\underline{v}(k)$ is measurement noise. The plant noise and measurement noise are assumed uncorrelated (white) with zero mean. That is,

$$E[w(k) \cdot w^T(\dot{J})] = Q'(k)\delta_{k\dot{J}}$$

and

$$E[v(k) \cdot v^T(\dot{J})] = R(k)\delta_{k\dot{J}}$$

with

$$\delta_{k\dot{J}} = 1 \quad , \quad k = \dot{J}$$
$$= 0 \quad , \quad k \neq \dot{J}$$

In order to apply the linear filter equation (3.2) is expanded in a Taylor series about the best estimate of the state at that time and only the first-order terms are kept. Equation (3.2) gives

$$z(k) = H(k) \cdot \underline{x}(k) + \underline{v}(k) \qquad (3.3)$$

where

$$H(k) = \frac{\partial h}{\partial \underline{x}}\Bigg|_{x'(k) = \hat{\underline{x}}(k/k-1)} \qquad (3.3a)$$

$\hat{\underline{x}}(k/k-1)$ is a predicted value of the state before the kth measurement.

A state error vector is defined by

$\tilde{\underline{x}}(k/k) = \hat{\underline{x}}(k/k) - \underline{x}(k)$,

and a predicted state error vector is defined by

$$\tilde{\underline{x}}(k/k-1) = \hat{\underline{x}}(k/k-1) - \underline{x}(k) .$$

The covariance of state error matrix is defined by

$$P(k/k) = E[\tilde{\underline{x}}(k/k) \cdot \tilde{\underline{x}}^T(k/k)] ,$$

and the predicted covariance of state error matrix is given by

$$P(k/k-1) = E[\tilde{\underline{x}}(k/k-1) \cdot \tilde{\underline{x}}^T(k/k-1)] .$$

The state excitation matrix is given by

$$Q(k) = \Gamma(k)\ E[\underline{w}(k) \cdot \underline{w}^T(k)] \cdot \Gamma^T(k)$$

and the measurement noise covariance matrix is

$$R(k) = E[\underline{v}(k) \cdot \underline{v}^T(k)] .$$

The Kalman filter equations are given by [1]:

$$P(k+1/k) = \Phi P(k/k)\Phi^T + Q(k) \tag{3.4a}$$

$$G(k) = P(k/k-1)H^T(k)[H(k)\cdot P(k/k-1)H^T(k) + R(k)]^{-1} \tag{3.4b}$$

$$P(k) = [I - G(k)H(k)]\ P(k/k-1) \tag{3.4c}$$

$$\hat{\underline{x}}(k+1/k) = \Phi\ \underline{x}(k/k) \tag{3.4d}$$

$$z(k/k-1) = \underline{h}(\underline{x}(k/k-1),\ k) \tag{3.4e}$$

$$\hat{x}(k) = \hat{x}(k/k-1) + G(k)[z(k) - \underline{z}(k/k-1)] \tag{3.4f}$$

The Q matrix serves not only to allow for maneuvering but also to account for any model inaccuracies, that is, any discrepancies between the true action of the torpedo and its characterization by Equation (3.1). The Q also serves to prevent the gain matrix G(k) from approaching zero by always insuring uncertainty in the predicted covariance of error matrix P(k+1/k).

B. OPTIMAL LINEAR SMOOTHING

Smoothing is a non-real-time data processing scheme that uses all measurements between 0 and T to estimate the state of a system at certain time t, where $0 \leq t \leq T$. The smoothed estimate of $\underline{x}(t)$ based on all the measurements between 0 and T is denoted by $\hat{\underline{x}}(t/T)$.

Smoother error covariance is denoted by P(t/T) and $P(t/T) \leq P(t)$ means that the smoothed estimate of $\underline{x}(t)$ is always better than or equal to its filtered estimate. This is shown graphically in Figure 2.

Several forms of the smoothing equations may be derived. One is the Rauch-Tung-Striebel form, which was used in our particular case with the discrete-time expressions summarized as follows [1]:

$$\hat{\underline{x}}(k/N) = \hat{\underline{x}}(k/k) + \underline{A}_k[\hat{\underline{x}}(k+1/N) - \hat{\underline{x}}(k+1/k)] \tag{3.5a}$$

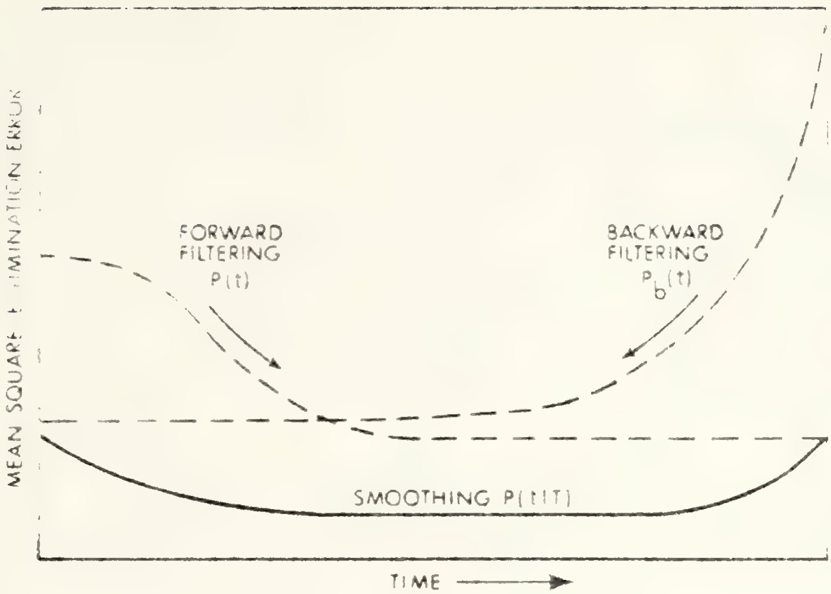


Figure 2. Advantage of Performing Optimal Smoothing [1]

where

$$\underline{A}_k = \underline{P}(k/k)\underline{\Phi}(k)^T\underline{P}(k+1/k)^{-1} \qquad \text{for } k = N-1$$

$$\underline{P}(k/N) = \underline{P}(k/k) + \underline{A}_k[\underline{P}(k+1/N) - \underline{P}(k+1/k)]\underline{A}_k^T \qquad (3.5b)$$

also for $k = N-1$.

In these equations $\underline{\hat{x}}(k/N)$ is smoothed State Estimate and $\underline{\hat{P}}(k/N)$ is Error Covariance Matrix Propagation.

IV. PROBLEM DEFINITION - TORPEDO TRACKING WITH THE EXTENDED KALMAN FILTER AND OPTIMAL SMOOTHING

A. FILTER EQUATIONS

In the torpedo tracking problem, the non-linear observations are the four independent transit times from the tracked object to the hydrophones, T_c, T_x, T_y and T_z. Thus the non-linear measurement matrix $\underline{z}(k)$ is defined as:

$$\underline{z}(k)=\begin{bmatrix} T_c(k) \\ T_x(k) \\ T_y(k) \\ T_z(k) \end{bmatrix}=\begin{bmatrix} \frac{1}{VEL}[x(k)+d/2)^2+(y(k)+d/2)^2+(z(k)+d/2)^2]^{1/2}+v(k) \\ \frac{1}{VEL}[(x(k)-d/2)^2+(y(k)+d/2)^2+(z(k)+d/2)^2]^{1/2}+v(k) \\ \frac{1}{VEL}[x(k)+d/2)^2+y(k)-d/2)^2+(z(k)+d/2)^2]^{1/2}+v(k) \\ \frac{1}{VEL}[(x(k)+d/2)^2+(y(k)+d/2)^2+(z(k)-d/2)^2]^{1/2}+v(k) \end{bmatrix} \tag{4.1}$$

The measurement noises, v(k)'s, are assumed to be zero-mean and independent with a covariance matrix

$$R(k) = \begin{bmatrix} \sigma^2_{T_c} & 0 & 0 & 0 \\ 0 & \sigma^2_{T_x} & 0 & 0 \\ 0 & 0 & \sigma^2_{T_y} & 0 \\ 0 & 0 & 0 & \sigma^2_{T_z} \end{bmatrix} \tag{4.2}$$

Equation (3.3a) can be used to give the linearized observation matrix. When the derivatives are taken and evaluated at the predicted state values $x(k/k-1) = x'(k)$ the result is

$$H(k)=\frac{1}{VEL}\begin{bmatrix} \frac{x'(k)+d/2}{DEN1} & 0 & \frac{y'(k)+d/2}{DEN1} & 0 & \frac{z'(k)+d/2}{DEN1} \\ \frac{x'(k)-d/2}{DEN2} & 0 & \frac{y'(k)+d/2}{DEN2} & 0 & \frac{z'(k)+d/2}{DEN2} \\ \frac{x'(k)+d/2}{DEN3} & 0 & \frac{y'(k)-d/2}{DEN3} & 0 & \frac{z'(k)+d/2}{DEN3} \\ \frac{x'(k)+d/2}{DEN4} & 0 & \frac{y'(k)+d/2}{DEN4} & 0 & \frac{z'(k)-d/2}{DEN4} \end{bmatrix} \qquad (4.3)$$

where

$$DEN1 = [(x'(k)+d/2)^2+(y'(k)+d/2)^2+(z'(k)+d/2)^2]^{1/2}$$

$$DEN2 = [(x'(k)-d/2)^2+(y'(k)+d/2)^2+(z'(k)+d/2)^2]^{1/2}$$

$$DEN3 = [(x'(k)+d/2)^2+(y'(k)-d/2)^2+(z'(k)+d/2)^2]^{1/2}$$

$$DEN4 = [(x'(k)+d/2)^2+(y'(k)+d/2)^2+(z'(k)-d/2)^2]^{1/2}$$

The torpedo dynamics used for the tracking problem are assumed to be $1/s^2$ with estimations on five states x position, x velocity, y position, y velocity and z position (height of torpedo above hydrophone array).

The means of the random excitation and random noise are assumed to be zero, i.e.,

$E[w(k)] = 0$

$E[v(k)] = 0$

Four measurements are taken every 1.31 seconds, which is one time slot, and with this sampling time the $1/s^2$ plant has state transition, (PHI) and gamma, (Γ) matrices equal to:

$$\underline{\Phi} = \begin{bmatrix} 1 & T & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & T & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

and

$$\underline{\Gamma} = \begin{bmatrix} T^2/2 & 0 & 0 \\ T & 0 & 0 \\ 0 & T^2/2 & 0 \\ 0 & T & 0 \\ 0 & 0 & T \end{bmatrix} \tag{4.5}$$

The Φ matrix, Q matrix, R matrix, and H matrix are then used in the Kalman filter equations (3.4).

B. THE SEQUENTIAL EXTENDED KALMAN FILTER

In the sequential approach, the basic Kalman filter equations (3.4) must be modified. Calculations are performed on each of the four independent transit times in the following order: T_c, T_x, T_y and T_z for each 1.31 second time slot. The estimate of the states x(k/k), based on one transit time measurement are used as the prediction x(k/k-1) for the calculations on the next measurement. Thus for the first time measurement T_c only the first row of the linearizing H matrix is calculated.

Next the first gain column corresponding to the first time measurement T_c is calculated by

$$G_{icol} = \frac{P(k/k-1)\, H^T_{irow}}{H_{irow} P(k/k-1)\, H^T_{irow} + R_{ii}} \tag{4.6}$$

where i = 1 to 4 corresponding to the four measured transit times. Thus, the first row of the H matrix is used to calculate the first column of the gain matrix with both corresponding to the first measured time T_c.

Next, an estimate of the particular observation time is calculated using equation (3.4f) evaluated at the predicted state $\underline{x}(k/k-1)$.

The difference between observed transit time and the estimated transit times forms the residual which is used in the estimate equation

$$x_i = x(k/k-1) + G_{icol}\,[\text{Residual}] \tag{4.7}$$

This equation gives an estimate of the states based on one of the four time measurements.

Next, covariance of error is calculated based on one measurement by

$$P_i = [I - G_{icol}\,H_{irow}]\,P_{i-1} \tag{4.8}$$

where

I = identity matrix

P_{i-1} = the covariance matrix calculated from the previous transit time measurement or if $i = 1$, the prediction $P(k/k-1)$.

After the first iteration, $\underline{x}_1$ becomes $\underline{x}(k/k-1)$ and P_1 becomes $P(k/k-1)$ for the second iteration which calculates the estimate of the states based on the second measurement T_x.

After four iterations ($k = 4$), $\underline{x}_4$ becomes the estimate for the time slot $x(k/k)$ and P_4 becomes the covariance error $P(k/k)$.

The predictions for the next time slot are calculated using equations (3.4a) and (3.4d). This process is repeated for each time slot.

C. OPTIMAL SMOOTHING PROCESS

During the running of the Extended Kalman filter and Smoothing routine, after the forward filter pass for each time slot (except the first), the smoothing subroutine is called. By using the present and previous filtered estimate of $\underline{x}(t)$, a smoothed estimate of previous $\underline{x}(t)$ is calculated. This process is repeated for each past time slot.

Solution of the equations (3.5) proceeds as follows: As an example, and because it is slightly easier to see when actual times are used, suppose N = 30. On the forward filter pass, the values $\hat{\underline{x}}(k/k)$, $\hat{\underline{x}}(k/k-1)$, $\underline{P}(k/k)$, and $\underline{P}(k/k-1)$ would be computed and stored. On the final iteration of the forward pass, with k = N = 30,

$$\hat{\underline{x}}(30/30) = \hat{\underline{x}}(30/29) + G(30)\ [\underline{z}(30) - \underline{H}\ \hat{\underline{x}}(30/29)]$$

i.e., we have computed and stored $\hat{\underline{x}}(30/30)$.

Now, the smoothing process starts in the reverse direction. Decrement k to k = N - 1 = 29, then

$$\hat{\underline{x}}(29/30) = \underbrace{\hat{\underline{x}}(29/29)}_{\text{stored}} + \underline{A}(29)\ [\underbrace{\hat{\underline{x}}(30/30)}_{\text{stored}} - \underbrace{\hat{\underline{x}}(30/29)}_{\text{stored}}]$$

and $$\underline{A}(29) = \underbrace{\underline{P}(29/29)}_{\text{stored}}\ \underline{\Phi}^T\ \underbrace{\underline{P}(30/29)}_{\text{stored}}{}^{-1}$$

Let $k = N - 2 = 28$, then

$$\underline{\hat{x}}(28/30) = \underbrace{\underline{\hat{x}}(28/28)}_{\text{stored}} + \underline{A}(28)\,[\underbrace{\underline{\hat{x}}(29/30)}_{\substack{\text{computed}\\\text{last}\\\text{iteration}}} - \underbrace{\underline{\hat{x}}(29/28)}_{\text{stored}}]$$

and $$\underline{A}(28) = \underbrace{\underline{P}(28/28)}_{\text{stored}}\ \underline{\Phi}^T\ \underbrace{\underline{P}(29/28)}_{\text{stored}}{}^{-1}$$

Also, for each of the two preceding iterations,

$$\underline{P}(29/30) = \underbrace{\underline{P}(29/29)}_{\text{stored}} + \underbrace{\underline{A}(29)}_{\text{computed}}[\underbrace{\underline{P}(30/30)}_{\text{stored}} - \underbrace{\underline{P}(30/29)}_{\text{stored}}]\ \underbrace{A^T(29)}_{\text{computed}}$$

$$\underline{P}(28/30) = \underbrace{\underline{P}(28/28)}_{\text{stored}} + \underbrace{\underline{A}(28)}_{\text{computed}}[\underbrace{\underline{P}(29/30)}_{\text{stored}} - \underbrace{\underline{P}(29/28)}_{\text{stored}}]\ \underbrace{A^T(28)}_{\text{computed}}$$

V. TESTING AND SIMULATION

A. DESCRIPTION

The sequential Extended Kalman Filter and Smoothing routine is tested using simulated torpedo tracks. A variety of track scenarios were produced to test the filter and smoothing performance during single and multiple arrays tracking.

Computer generated tracks were tested in the first series of straight running, constant depth and constant velocity torpedoes. A variety of track scenarios were used transiting through multiple quadrants including:

1. Crossing north of the array.
2. Crossing diagonally through the array.

The next series of tests demonstrates the ability of the filter to track through the areas of multiple arrays including:

1. Crossing above the arrays.
2. Crossing diagonally through the arrays.

All runs were made with a variety of initialization errors in position and velocity.

Zero mean Gaussian noise is added to corrupt the observed transit times for all runs.

B. THE GATING SCHEME

The operation of the filter may be adversely affected by large measurement noise. One error of a relatively large magnitude could invalidate the filtered output for many subsequent time slots. Before random measurement noise and random excitations could be added to the observed times for testing, a form of protection was designed to guard against catastrophic failure. This protection is provided by establishing limits of acceptability for each of the measurements.

Measurement errors can occur because of many factors including an error in the transit time of the acoustic pulse primarily due to the receipt of multipath signals that have bounced off the surface, bottom or different density layers.

A three-sigma gate was designed using the covariance of measurement noise (R) and the covariance of estimation error (P(k/k)).

For each calculation of a state estimate ($\underset{\sim}{x}(k/k)$), the largest positional covariance of error was used, either x, y or z, and converted to time in seconds using the average velocity of sound in water for Dabob Bay, 4860 ft/sec. The gate then was written for each time measurement i = 1 to 4:

$$\text{GATE} = \sqrt{\frac{P(k/k)_{\text{largest}}}{(4860.)^2} + R_{ii}}$$

The gate expands or decreases depending on the confidence level of the position estimate and the transit time. If ZDIFF, which is the difference between the actual transit time received and the predicted transit time to a particular hydrophone, exceeds the gate, the measurement is considered unacceptable and the filter gain is set to zero causing the filter to ignore the data and take the prediction of the states as the estimate

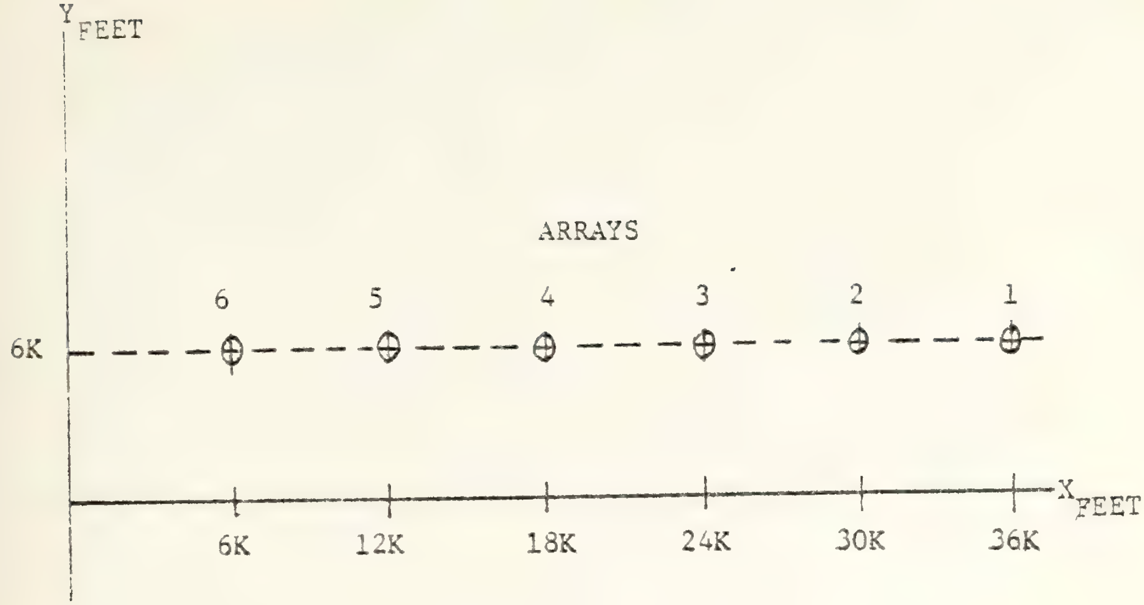
$$\underline{x}(k/k) = \underline{x}(k/k-1)$$

An invalid time measurement zeros only the gain column for that particular hydrophone causing only that hydrophone's data to be ignored.

C. MULTIPLE ARRAY TRACKING

Initial tests were performed on tracks in the area of one array. In order to more closely simulate a typical run on the range, a scheme was designed to track the torpedo through multiple arrays.

First, a coordinate system is defined as shown in Figure 3. The center of the coordinate system is geographically near the entrance to Dabob Bay in the simulation. Array number 6 is the closest array to be coordinate center. In the simulation array 1 is at 36,000 feet from coordinate center and array 6 is 6,000 feet. The C hydrophone is assumed to be the axis location of each array. Then each X



Coordinate System for Multiple Array Tracking

	C HYDRO			X HYDRO			V HYDRO			Z HYDRO		
	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z
ARRAYS	36000	6000	0	36030	6000	0	36000	6030	0	36000	6000	30
	30000	6000	0	30030	6000	0	30000	6030	0	30000	6000	30
	24000	6000	0	24030	6000	0	24000	6030	0	24000	6000	30
	18000	6000	0	18030	6000	0	18000	6030	0	18000	6000	30
	12000	6000	0	12030	6000	0	12000	6030	0	12000	6000	30
	6000	6000	0	6030	6000	0	6000	6030	0	6000	6000	30

HYDRO--Hydrophone Location Matrix

Figure 3

position for the X hydrophone in each array is $X_c + 30$, each Y position for the Y hydrophone is $Y_c + 30$, and each Z position for the Z hydrophone is $Z_c + 30$. These 72 positions, an XYZ position for each of 4 hydrophones in 6 arrays, were placed into a 6 x 12 matrix HYDRO and referenced throughout the routine.

The geometry centered on each array is taken out of the problem and the target position is based on a central reference.

The non-linear time equation becomes

$$T = 1/VEL\sqrt{(x - x_o)^2 + (y - y_o)^2 + (z - z_o)^2}$$

where x_o, y_o, or z_o is the position of a particular hydrophone and array being used.

The decision parameter used to determine the switching from array to array is a straight handoff. If the predicted x position is greater than 3,000 feet from the array in use, then an index (I8) is incremented and the next row of HYDRO is implemented. This placed into the routine the x, y and z positions of the hydrophones in the next array. The handoff can easily be utilized in real range operations, as the transit times from adjacent arrays are present at the computer for a particular time slot. For simulation, it is assumed that in all the arrays each axis pointed in the same direction. In actual range operations each array is tilted

about both the X and Y axis. Since the true transit times are derived in a tilted coordinate frame, the filter's estimate of transit time must also be calculated in a tilted coordinate frame. The tilt angle measurements along with the level rectangular coordinates of the array with respect to the central rectangular coordinate system can be input into the matrix HYDRO to rotate the coordinates of each hydrophone in the array.

VI. TEST RESULTS

A. SERIES ONE

Figure 5 shows the true trajectory of the torpedo in the horizontal X-Y plane during a straight run through single array. Torpedo velocity is 50 knots in the x-direction. Initial position errors are set to 25 feet for X and Y. Velocity errors are set to zero. Figures 6, 7 and 8 depict the position errors for both Kalman filter and Smoothing. Measurement noise is added to all runs. The steady state X and Y position errors ranged between -6 and +9 feet throughout the trajectory for Kalman filter and -2 and +4 feet for smoothing. The position errors are computed by subtracting the filter position estimate, $x(k/k)$, for Kalman filter and $x(k/N)$, for smoothing, from the computer generated true position for each time slot. Figures 9, 10, 11, 12, and 13 depict the mean square estimation errors of states. These estimation errors are obtained by taking the appropriate diagonal terms of the covariance matrix and smoothed covariance matrix.

B. SERIES TWO

Figure 14 shows the true trajectory of the torpedo in the horizontal X-Y plane, during a crossing run through single array. Torpedo velocity is 40 knots in X-direction

and 25 knots in Y-direction. The torpedo depth is maintained at 300 feet. Figures 15, 16, and 17 depict the position errors. Since the initial position errors are set to zero, the position errors ranged between -3 and +4 feet. Figures 18, 19, 20, 21, and 22 depict the mean square estimation errors of states.

C. SERIES THREE

Figure 23 shows the true trajectory of the torpedo in the horizontal X-Y plane, during a straight run through multiple array. Because of storage problem of the computer, the runs through the multiple array were made for 190 time slots. Torpedo velocity is 50 knots in X-direction and the depth is 100 feet. Figures 24, 25, and 26 show the position errors ranged between -6 and 17 feet. Figures 27, 28, 29, 30, and 31 depict the mean square estimation errors of states. Figure 32 shows the error ellipsoids superimposed at every eighteenth observation. The error ellipsoids are expanded to twenty-five times their true value in order that they may be seen. The error ellipsoids provide a geometric interpretation of the behavior of the estimator. Before the hand-off point, at the ninetyth time slot, the major axis rotation of the error ellipsoid and magnitude of the axis were -16.98 degrees and 43.28 feet, respectively. After the hand-off point major axis rotation became 25.8 degrees, and its magnitude became 1.6 feet. When the magnitude of an

axis of the ellipsoid decreases, the conclusion is that the error in the estimate decreases, because the observation from the new array has an error covariance ellipse rotated over 40 degrees from the present covariance (-16.94 degree) of the estimate. The ellipsoids are extremely narrow. When combined the resultant covariance is reduced greatly. Figure 4 depicts the result of reduction and rotation of the ellipsoids. Figure 33 shows the error ellipsoids before and after the hand-off point.

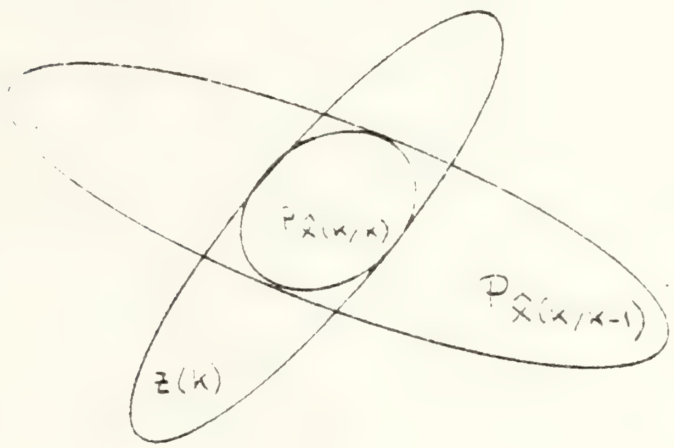


Figure 4. Result of Rotation and Reduction of the Error Ellipsoids

D. SERIES FOUR

Figure 34 shows the true trajectory of the torpedo in the horizontal X-Y plane, during a crossing run through multiple array. Torpedo velocity is 50 knots in X-direction and 40 knots in Y-direction. The torpedo depth is maintained at 300 feet. Initial position errors are set to 25 feet for X and Y and initial velocity errors are set to 5 knots. Figures 35, 26, and 37 show the position and depth errors. Since the initial position and velocity errors are set to 25 feet and 5 knots, the big position errors were taken at the beginning of the run. These values were ignored from the figures in order to see clearly to the rest of the run. Figures 38, 39, 40, 41, and 42 show the mean square estimation errors of states for both filtered and smoothed.

VII. CONCLUSIONS

The sequential Extended Kalman Filter and Smoothing satisfactorily provided real time estimates of torpedo position and depth. The average of steady state position and depth errors ranged between 3 and 1 feet for torpedo tracks within the specified radial tracking range after Kalman filter. These errors had a range of around 1 foot after smoothing.

The filter performance was dependent on system noise and the distance the torpedo was from the hydrophone array and the smoothed estimates of states were better than or equal to the filter estimates.

Implementation at the range computer facilities can be accomplished by real time Kalman Filtering and post run Smoothing of the raw time data. Future tests should include evaluating filter performance using trajectories generated from actual torpedo runs on the Dabob test range. These tests would verify the adequacy of the noise model in the filter and the ability of the software to edit erroneous transit time measurements.

The rotation and reduction of the error ellipsoids (i.e., the filter error covariance) was most instructive and gave much insight into the performance of the filter.

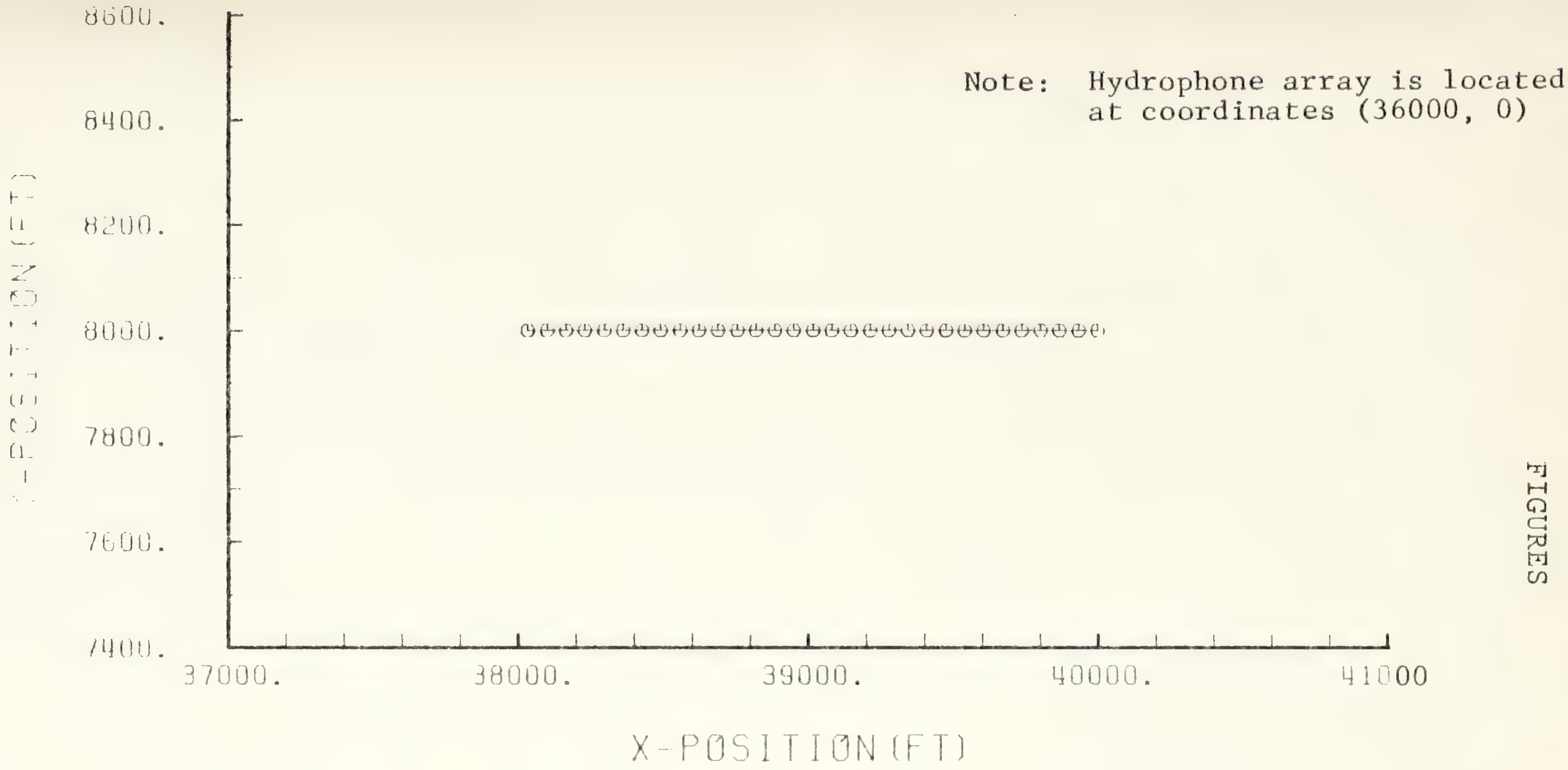


Figure 5. True Trajectory of the Torpedo During a Straight Run Through Single Array (Right to Left)

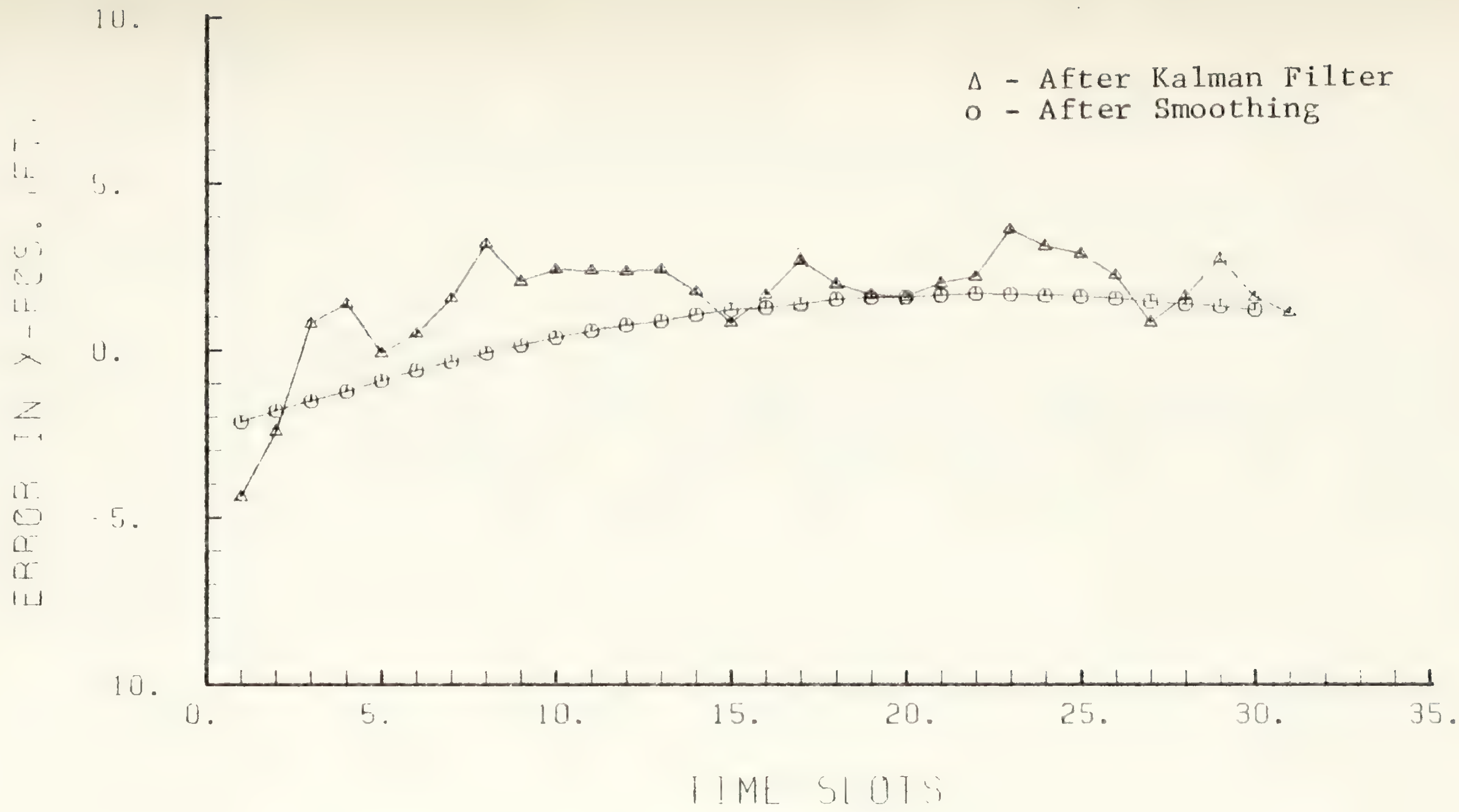


Figure 6. Error in Torpedo X-Position During a Straight Run Through Single Array

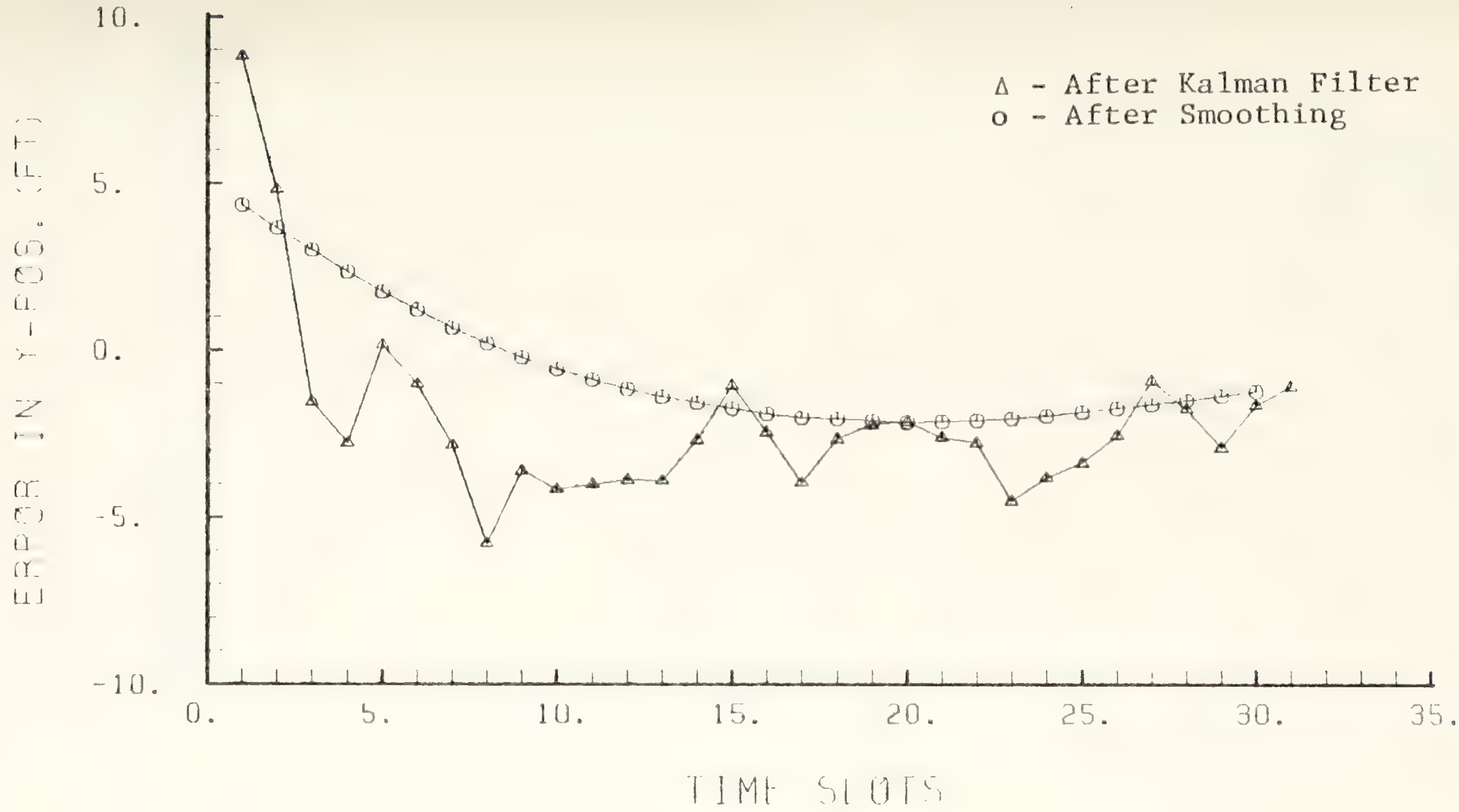


Figure 7. Error in Torpedo Y-Position During a Straight Run Through Single Array

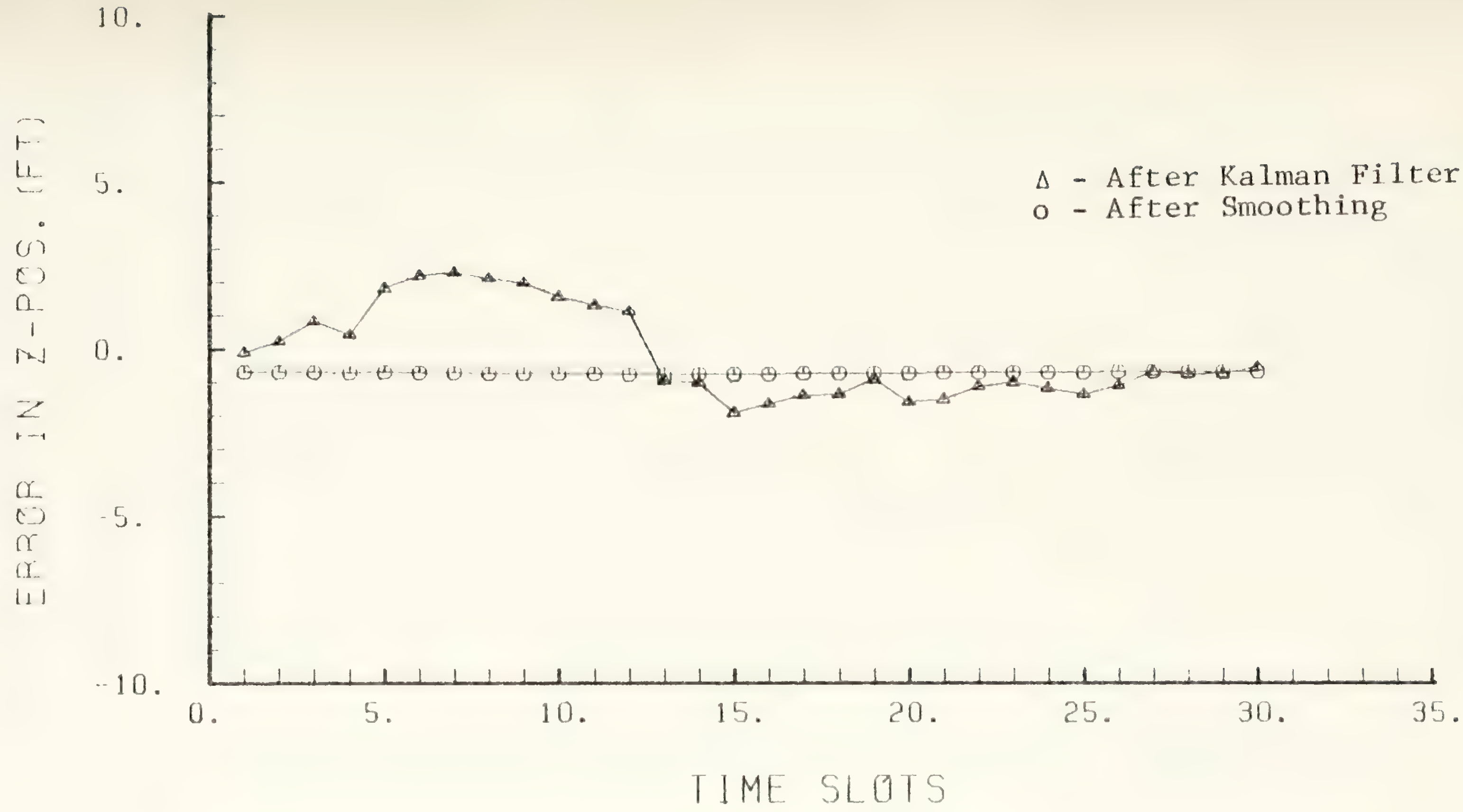


Figure 8. Error in Torpedo Z-Position During a Straight Run Through Single Array

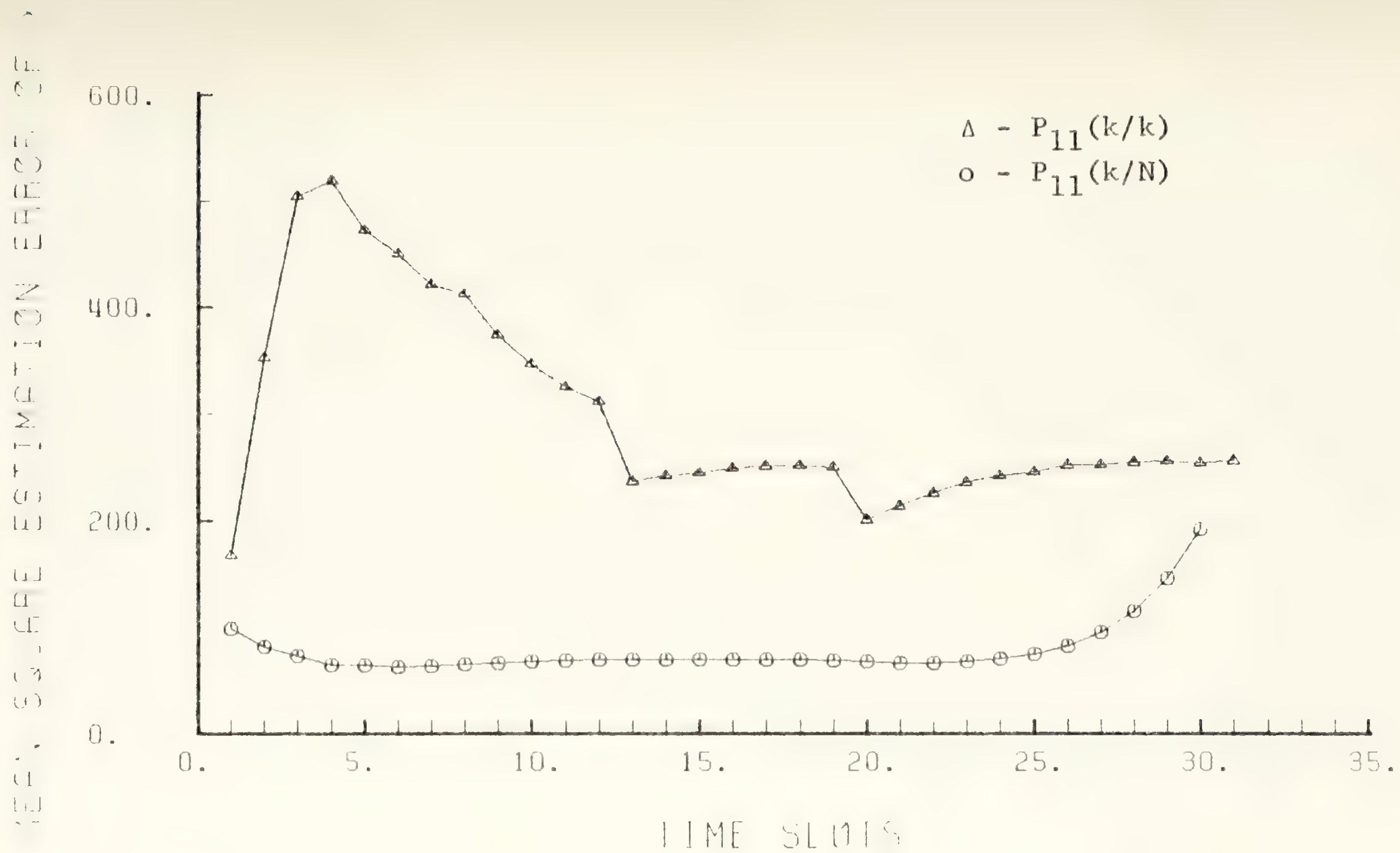


Figure 9. Mean Square Estimation Error (ft.^{2}) in X-Position During a Straight Run Through Single Array

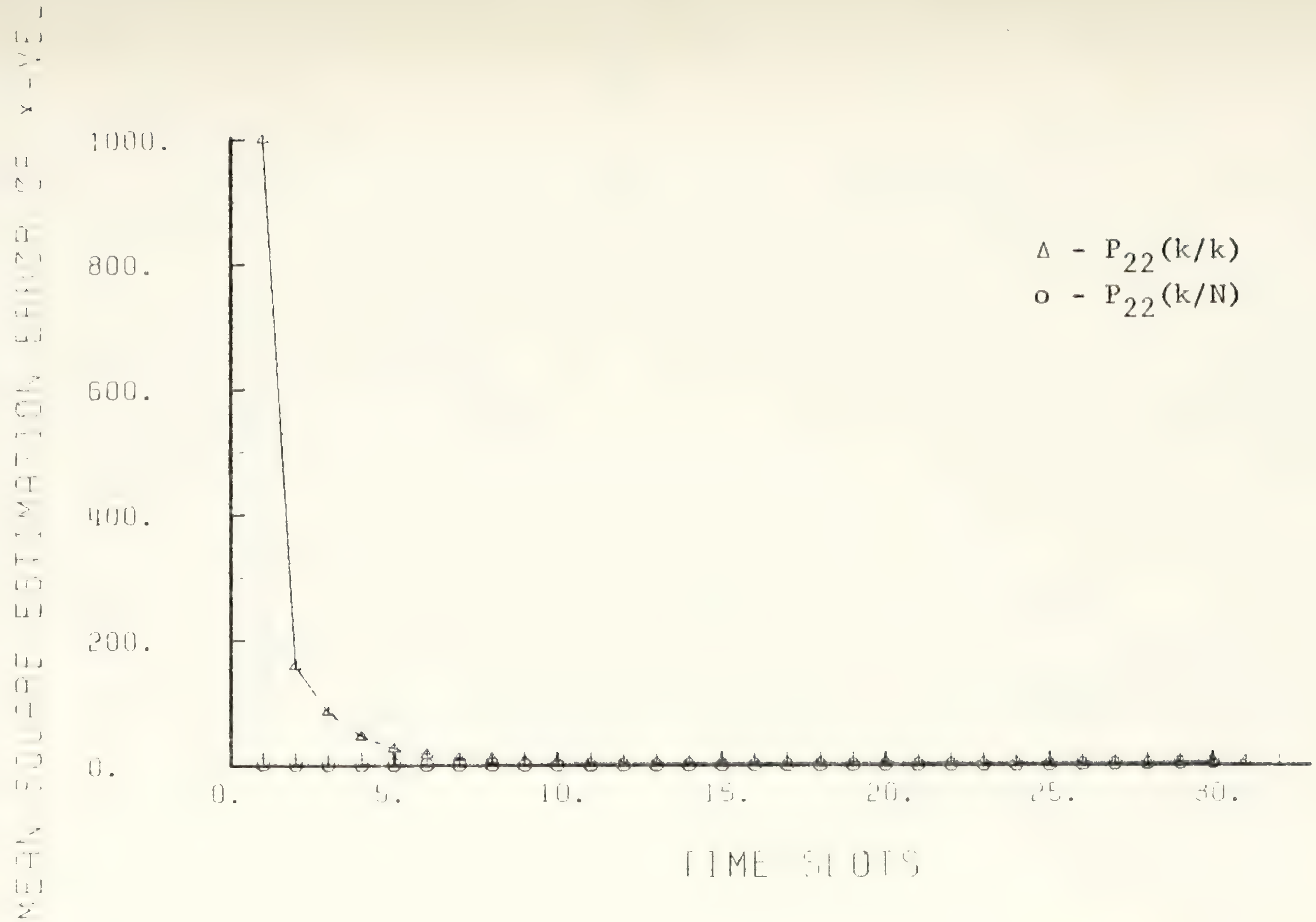


Figure 10. Mean Square Estimation Error (ft.^{2}/sec.^{2}) in X-Velocity During a Straight Run Through Single Array

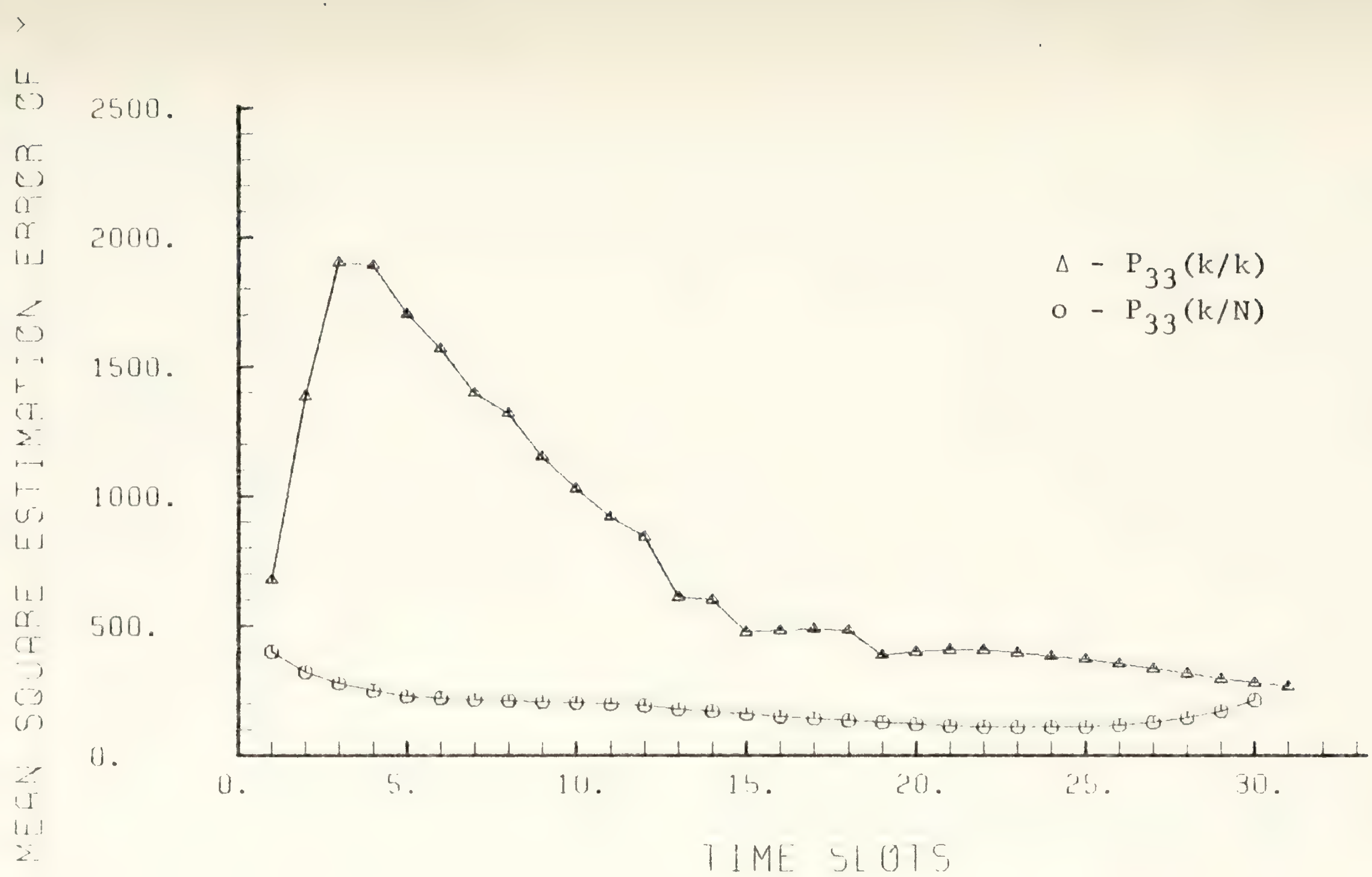


Figure 11. Mean Square Estimation Error (ft.^{2}) in Y-Position During a Straight Run Through Single Array

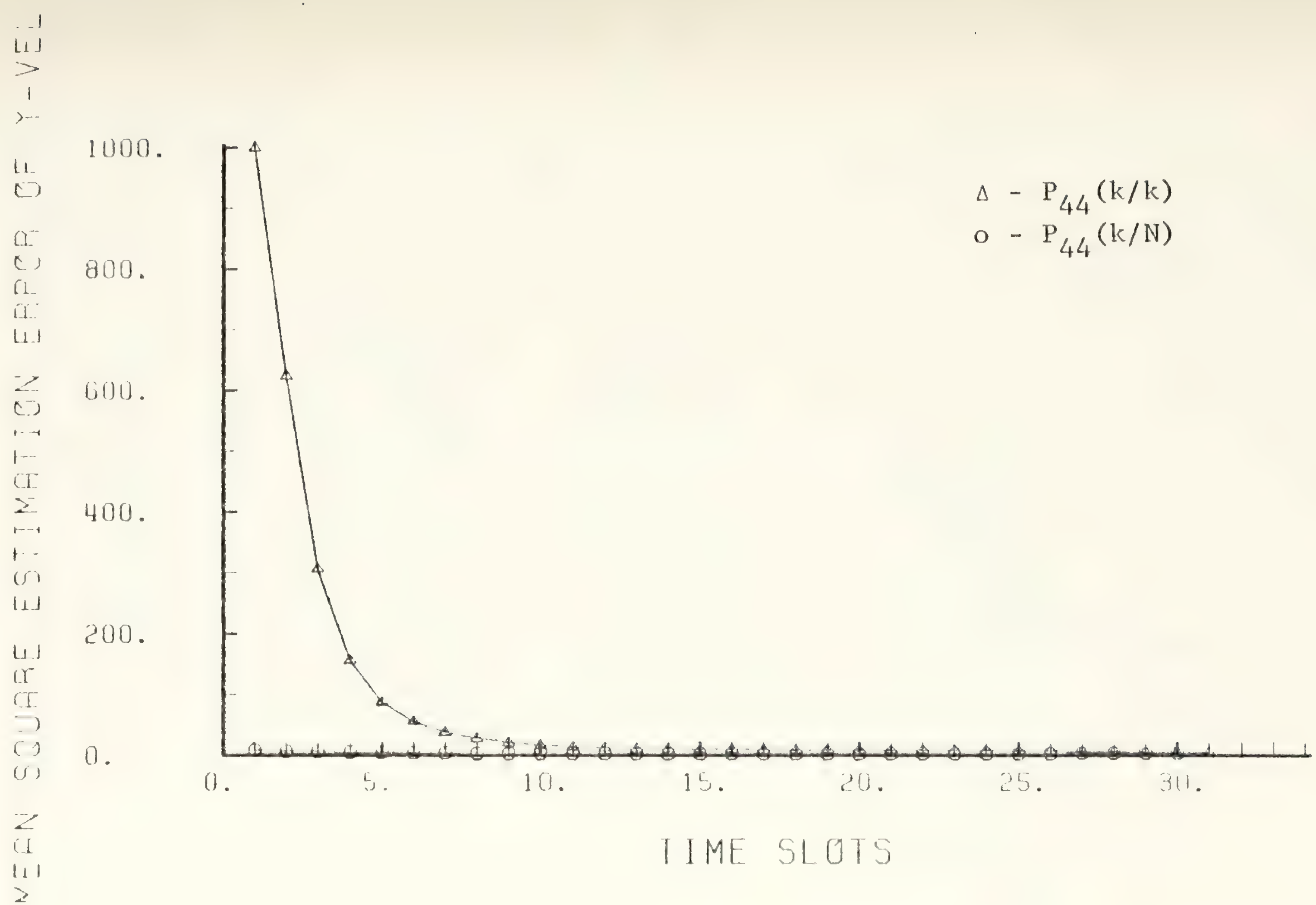


Figure 12. Mean Square Estimation Error (ft.^{2}/sec.^{2}) in Y-Velocity During a Straight Run Through Single Array

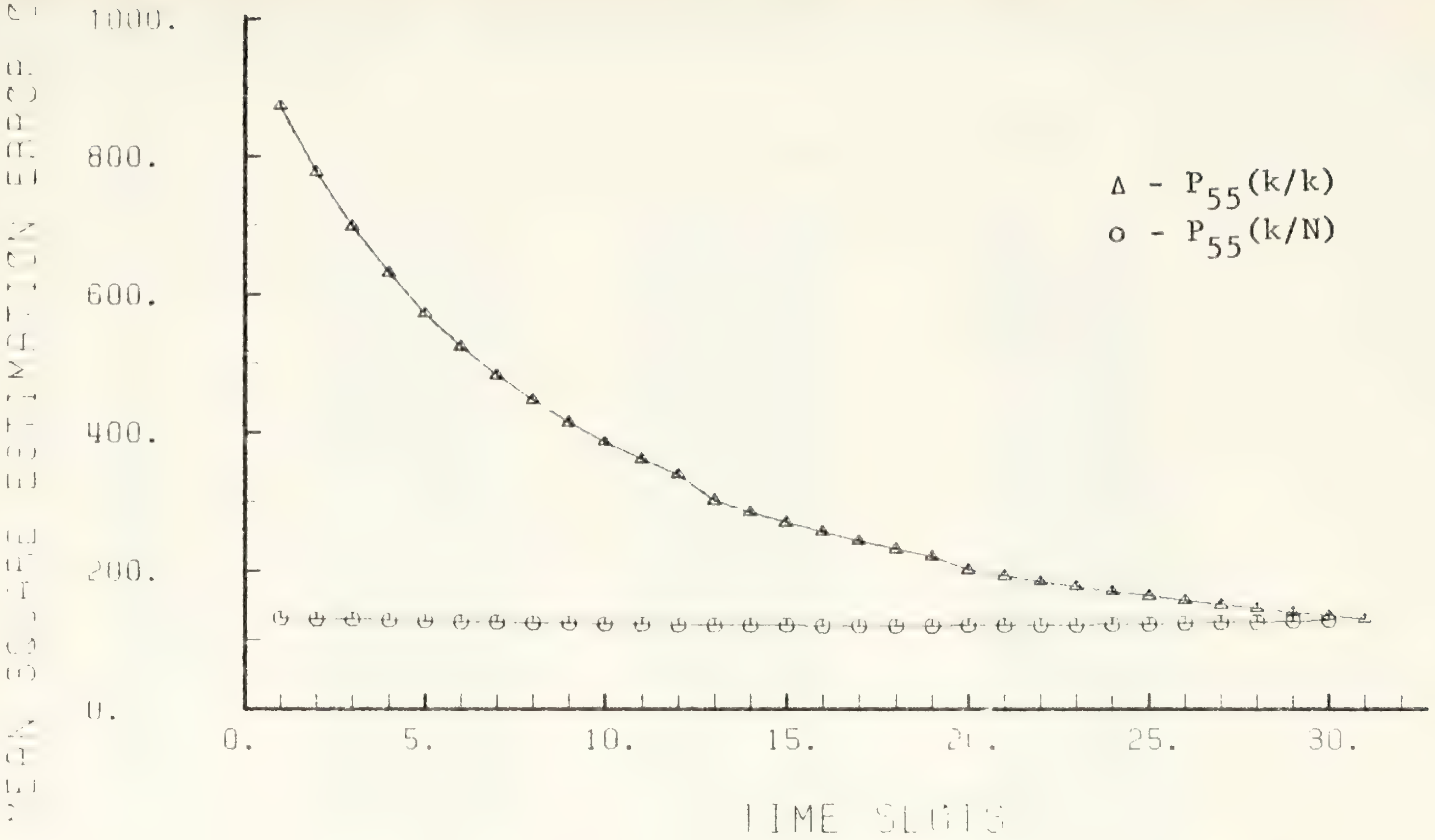


Figure 13. Mean Square Estimation Error (ft.^{2}) in Z-Position During a Straight Run Through Single Array

TABLE 1

X-Position (ft.) Results of Straight Run Through Single Array

TIME	TRUE-X	X-AFTER KALMAN	X-AFTER SMOOTHING
1	40000.0000	40004.3750	40002.1367
2	39934.5000	39936.9297	39936.3125
3	39869.0000	39868.2031	39870.5156
4	39803.5000	39802.1016	39804.7383
5	39738.0000	39738.0977	39738.9102
6	39672.5000	39672.0000	39673.1055
7	39607.0000	39605.4492	39607.3359
8	39541.5000	39538.3164	39541.5937
9	39476.0000	39473.9570	39475.8516
10	39410.5000	39408.0937	39410.1250
11	39345.0000	39342.6172	39344.4297
12	39279.5000	39277.1523	39278.7656
13	39214.0000	39211.5937	39213.1289
14	39148.5000	39146.7461	39147.4531
15	39083.0000	39082.1953	39081.8086
16	39017.5000	39015.8594	39016.2422
17	38952.0000	38949.3164	38950.6367
18	38886.5000	38884.5391	38884.9922
19	38821.0000	38819.3555	38819.4336
20	38755.5000	38753.9141	38753.9219
21	38690.0000	38688.0039	38688.3711
22	38624.5000	38622.3086	38622.8281
23	38559.0000	38555.3867	38557.3396
24	38493.5000	38490.3984	38491.8750
25	38428.0000	38425.1250	38426.4023
26	38362.5000	38360.2617	38360.9609
27	38297.0000	38296.1836	38295.5430
28	38231.5000	38229.8984	38230.1172
29	38166.0000	38163.2695	38164.6992
30	38100.5000	38098.9062	38099.2891

TABLE 2

Y-Position (ft.) Results of Straight Run Through Single Array

TIME	TRUE-Y	Y-AFTER KALMAN	Y-AFTER SMOOTHING
1	8000.0000	7991.1758	7995.6562
2	8000.0000	7995.1875	7996.3359
3	8000.0000	8001.5508	7997.0117
4	8000.0000	8002.7852	7997.6523
5	8000.0000	7999.8516	7998.2578
6	8000.0000	8001.0273	7998.8086
7	8000.0000	8002.8359	7999.3320
8	8000.0000	8005.7891	7999.8047
9	8000.0000	8003.6133	8000.2148
10	8000.0000	8004.1641	8000.5742
11	8000.0000	8004.0234	8000.8906
12	8000.0000	8003.8906	8001.1680
13	8000.0000	8003.9023	8001.4102
14	8000.0000	8002.6914	8001.5820
15	8000.0000	8001.0703	8001.7383
16	8000.0000	8002.4609	8001.9102
17	8000.0000	8003.9609	8002.0156
18	8000.0000	8002.6680	8002.0547
19	8000.0000	8002.2305	8002.1172
20	8000.0000	8002.1562	8002.1680
21	8000.0000	8002.6211	8002.1523
22	8000.0000	8002.7734	8002.1016
23	8000.0000	8004.5273	8002.0508
24	8000.0000	8003.8164	8001.9766
25	8000.0000	8003.3750	8001.8633
26	8000.0000	8002.5586	8001.7500
27	8000.0000	8000.9180	8001.6406
28	8000.0000	8001.7578	8001.5117
29	8000.0000	8002.9258	8001.3750
30	8000.0000	8001.6562	8001.2344

TABLE 3

Z-Position (ft.) Results of Straight Run Through Single Array

TIME	TRUE-Z	Z-AFTER KALMAN	Z-AFTER SMOOTHING
1	100.0000	100.1117	100.6952
2	100.0000	99.7705	100.6971
3	100.0000	99.1682	100.6939
4	100.0000	99.5596	100.7045
5	100.0000	98.1741	100.6970
6	100.0000	97.8086	100.7122
7	100.0000	97.7114	100.7122
8	100.0000	97.8788	100.7305
9	100.0000	98.0196	100.7323
10	100.0000	98.4320	100.7464
11	100.0000	98.6859	100.7537
12	100.0000	98.8770	100.7609
13	100.0000	100.9539	100.7732
14	100.0000	101.0273	100.7662
15	100.0000	101.9257	100.7623
16	100.0000	101.6473	100.7589
17	100.0000	101.3949	100.7525
18	100.0000	101.3649	100.7401
19	100.0000	100.9173	100.7383
20	100.0000	101.5895	100.7365
21	100.0000	101.5225	100.7264
22	100.0000	101.1333	100.7166
23	100.0000	101.0129	100.7104
24	100.0000	101.1896	100.7113
25	100.0000	101.3688	100.7001
26	100.0000	101.1070	100.6941
27	100.0000	100.6891	100.6891
28	100.0000	100.7637	100.6861
29	100.0000	100.7554	100.6837
30	100.0000	100.5455	100.6821

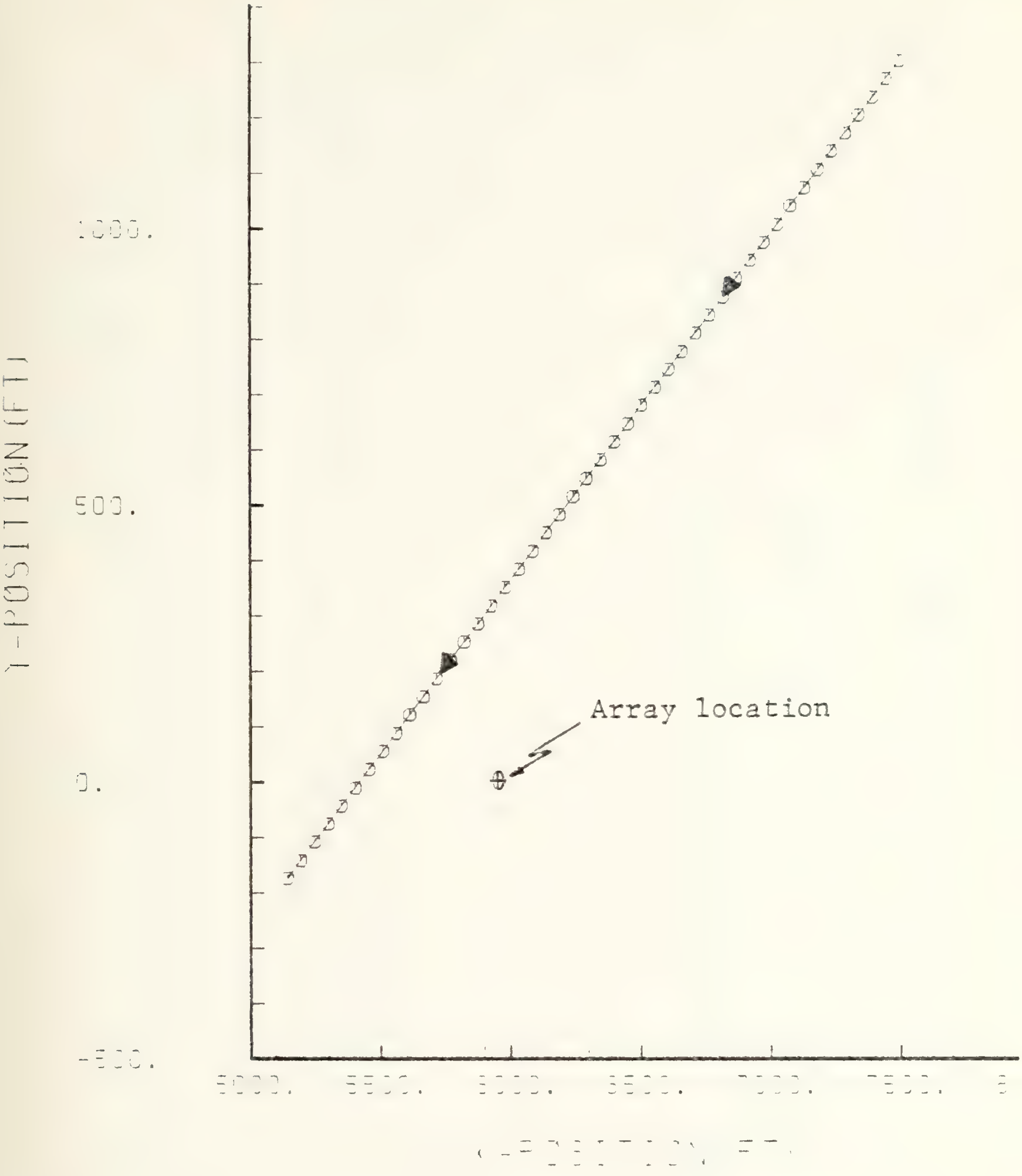


Figure 14. True Trajectory of the Torpedo During a Straight Run Through Single Array

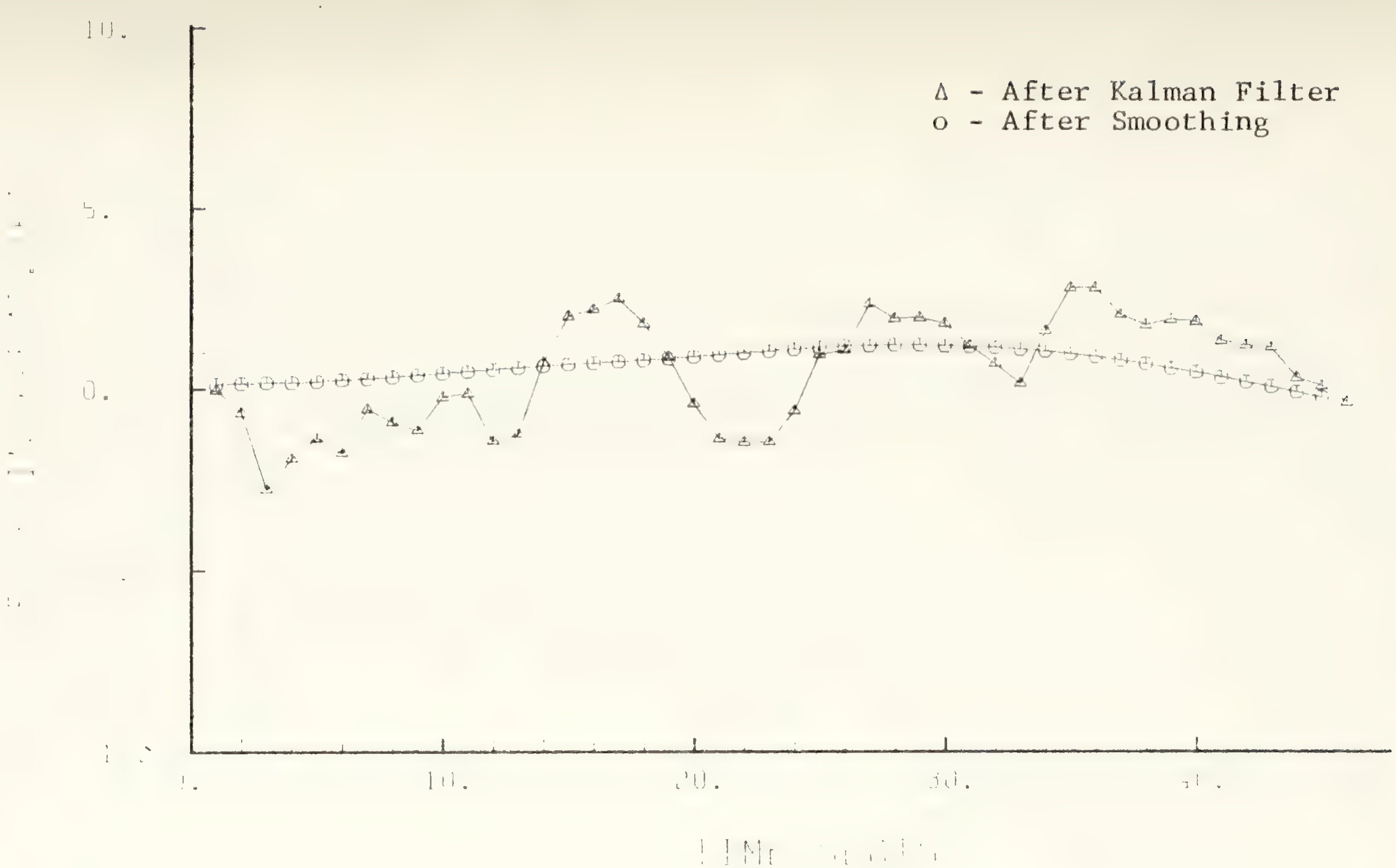


Figure 15. Error in Torpedo X-Position During a Straight Run Through Single Array

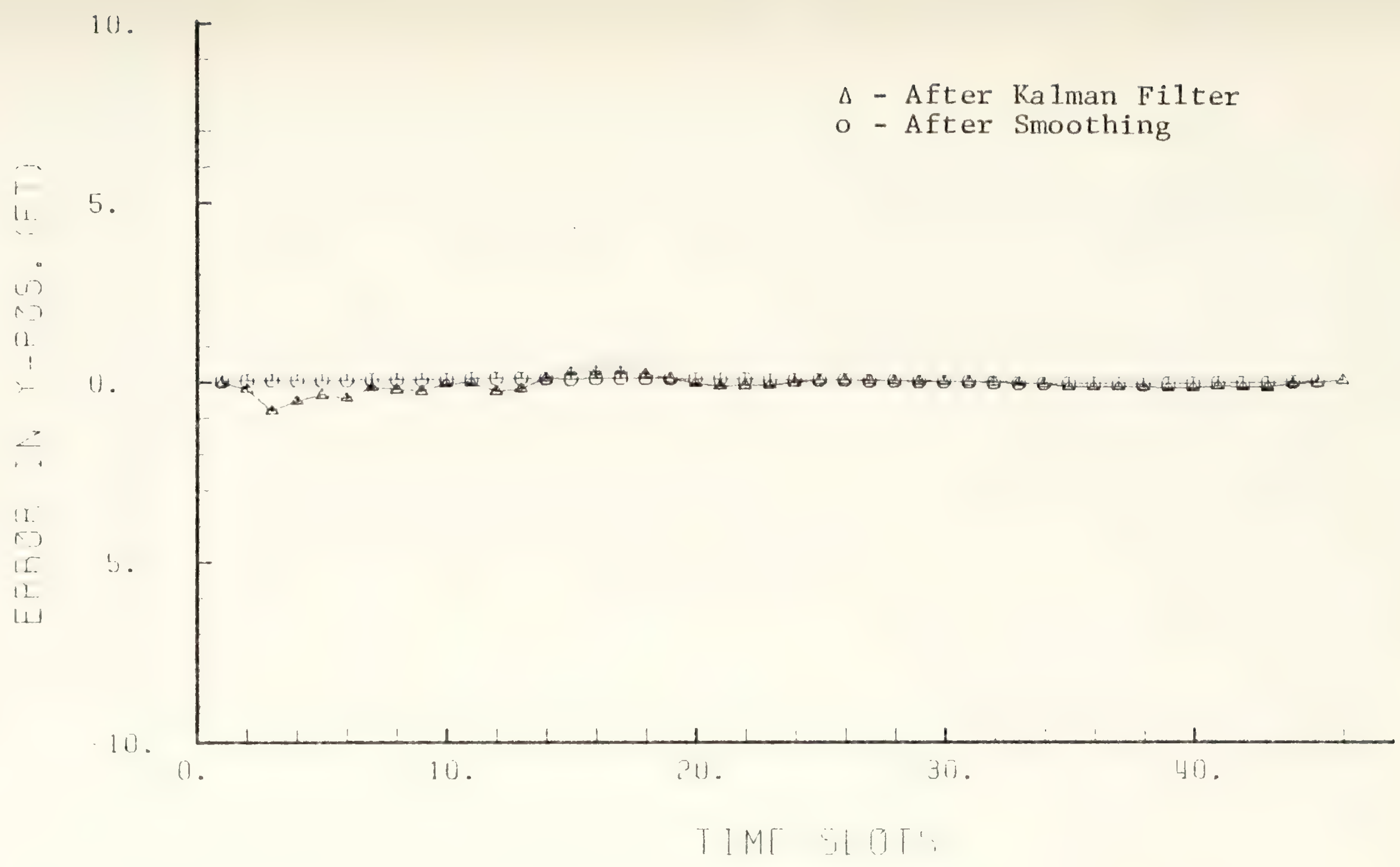


Figure 16. Error in Torpedo Y-Position During a Straight Run Through Single Array

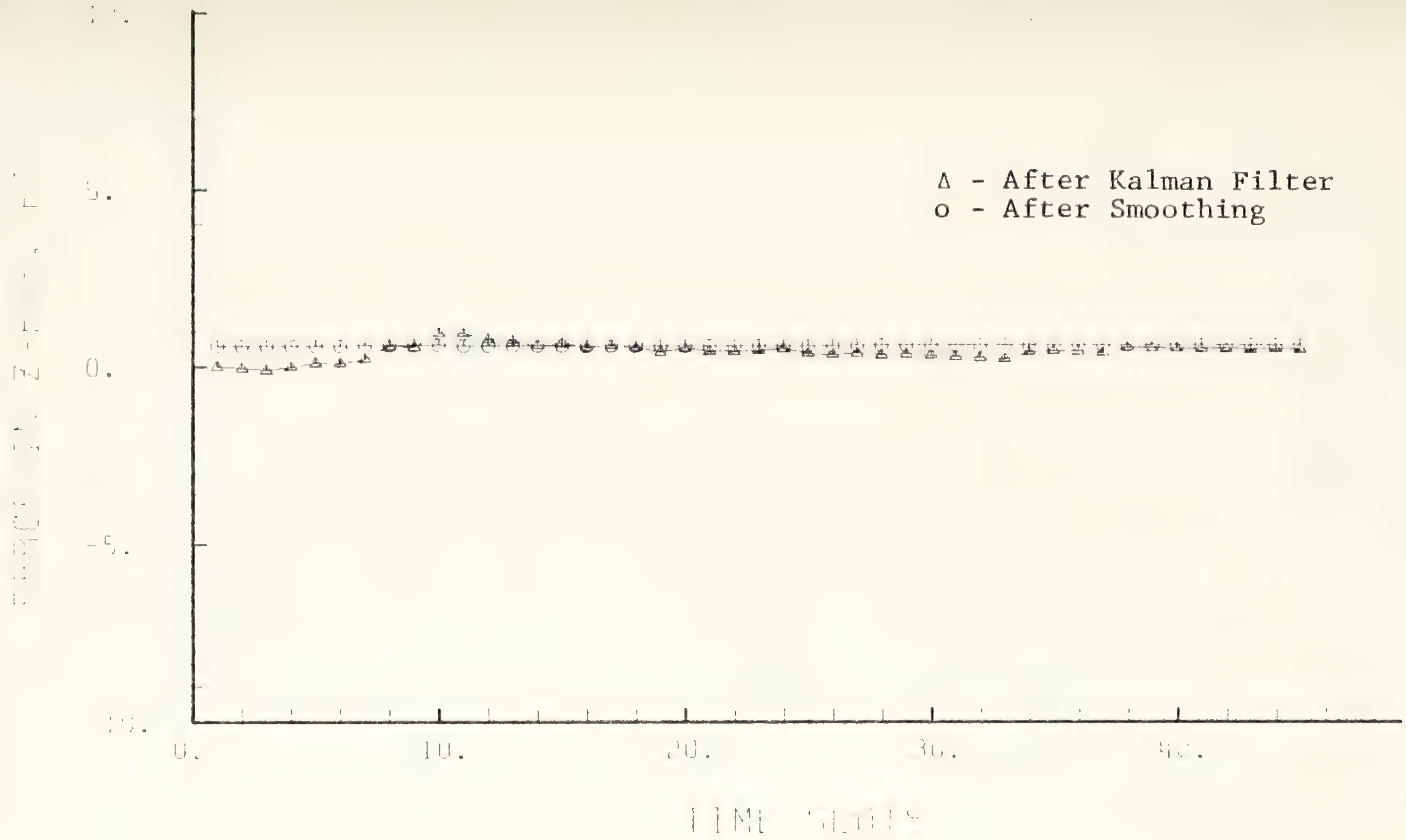


Figure 17. Error in Torpedo Z-Position During a Straight Run Through Single Array

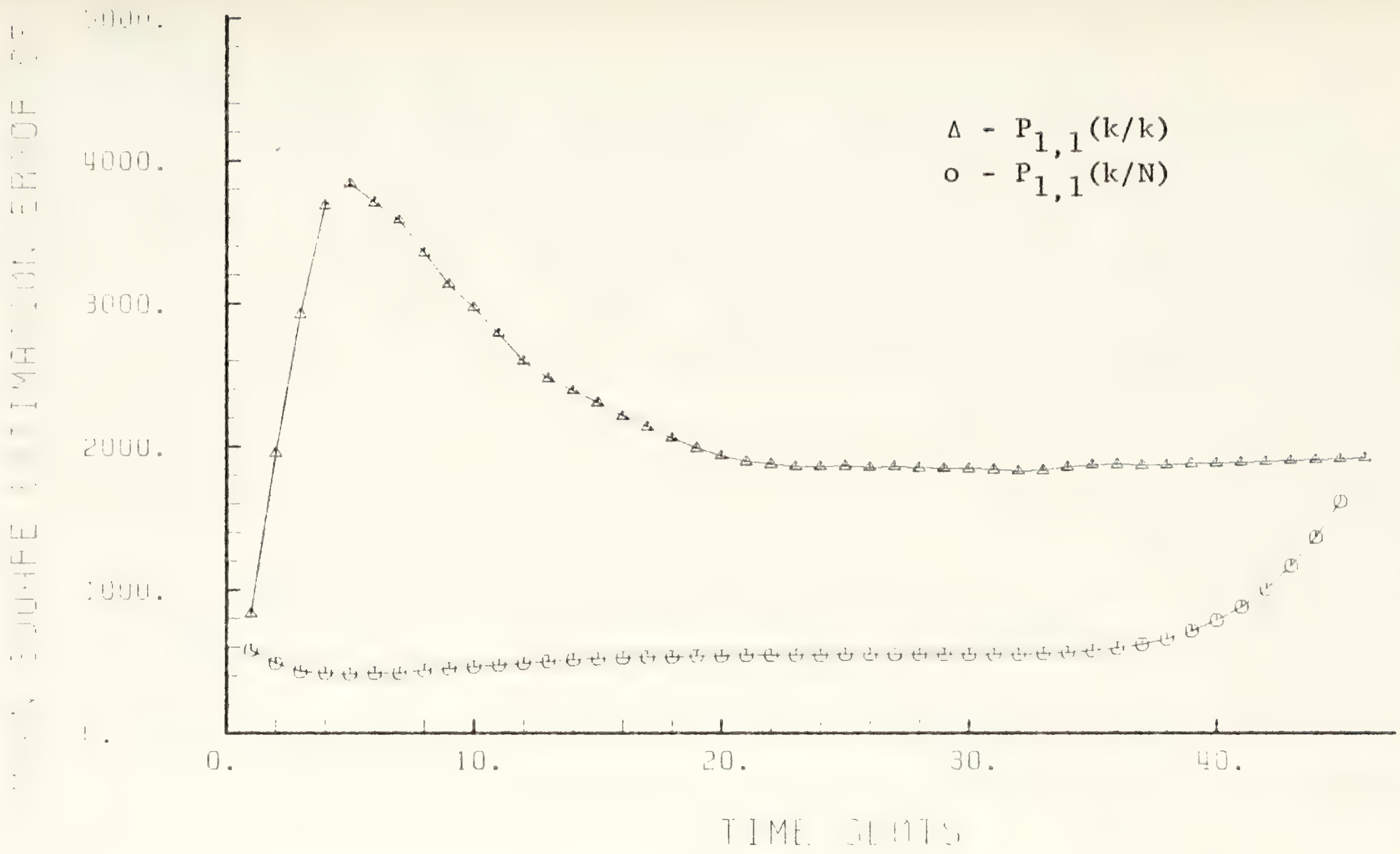


Figure 18. Mean Square Estimation Error (ft.^{2}) in X-Position During a Straight Run Through Single Array

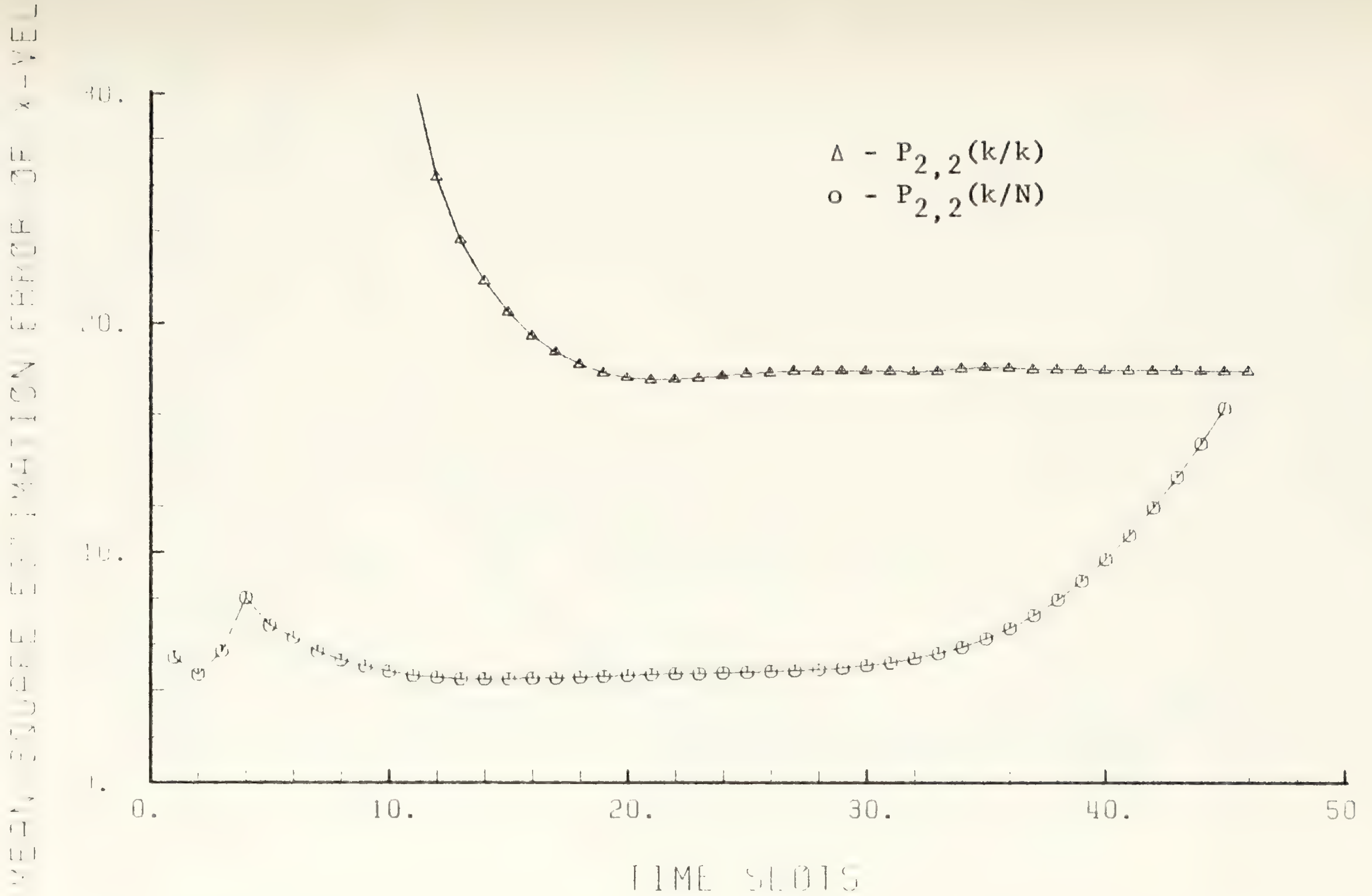


Figure 19. Mean Square Estimation Error (ft.^{2}/sec.^{2}) in X-Velocity During a Straight Run Through Single Array

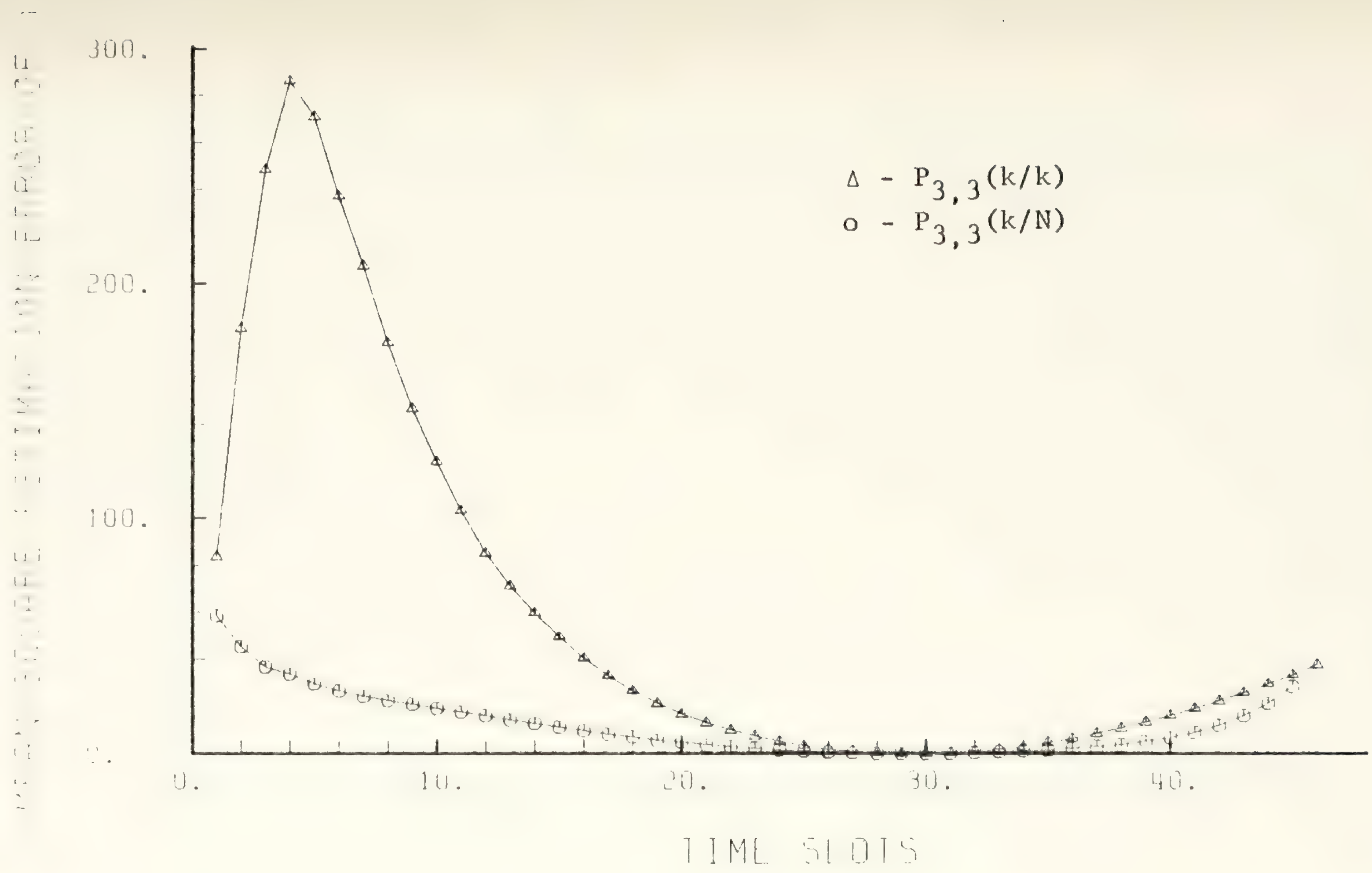


Figure 20. Mean Square Estimation Error (ft.^{2}) in Y-Position During a Straight Run Through Single Array

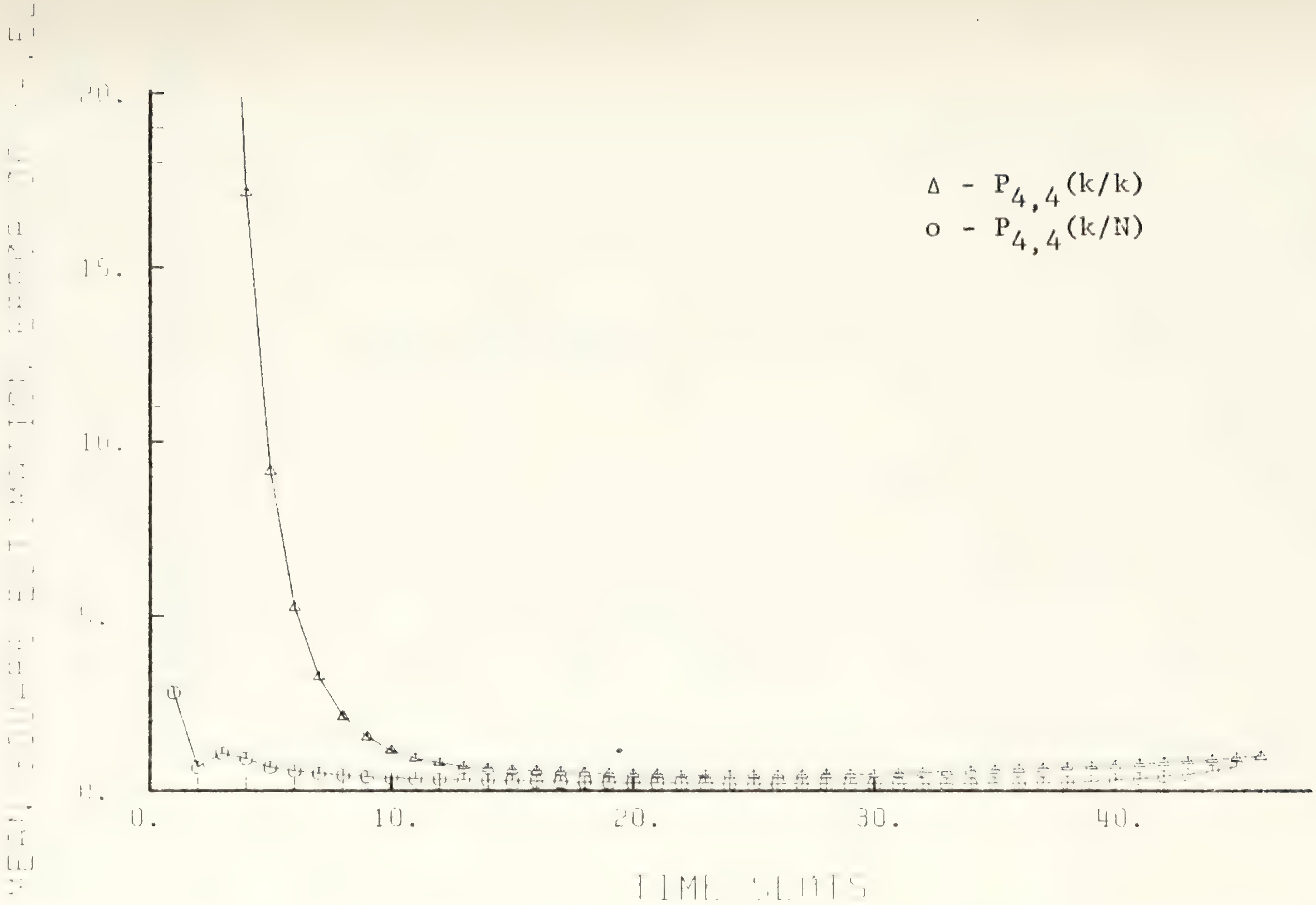


Figure 21. Mean Square Estimation Error (ft.^{2}/sec.^{2}) in Y-Velocity During a Straight Run Through Single Array

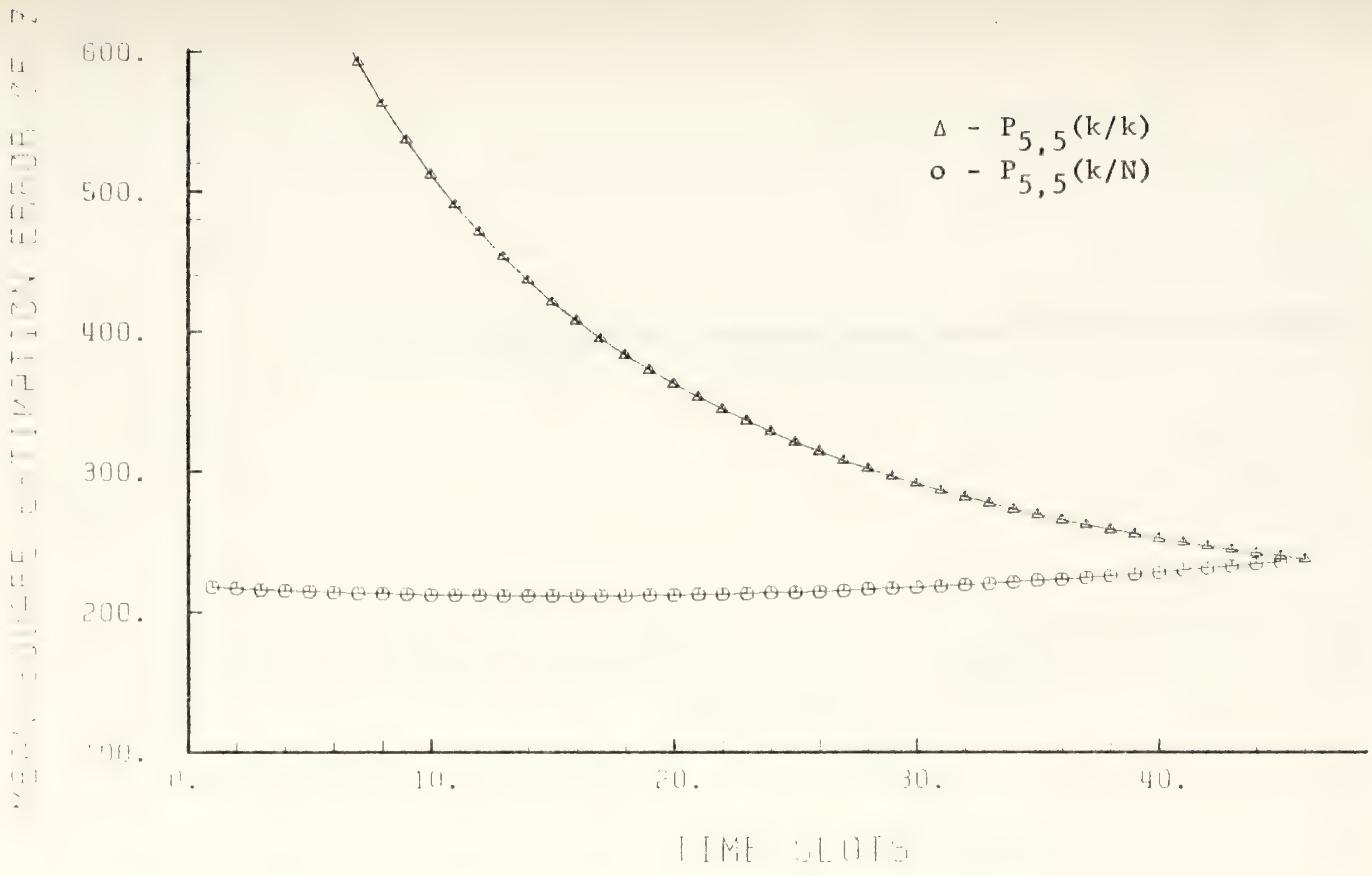


Figure 22. Mean Square Estimation Error (ft.^{2}) in Z-Position During a Straight Run Through Single Array

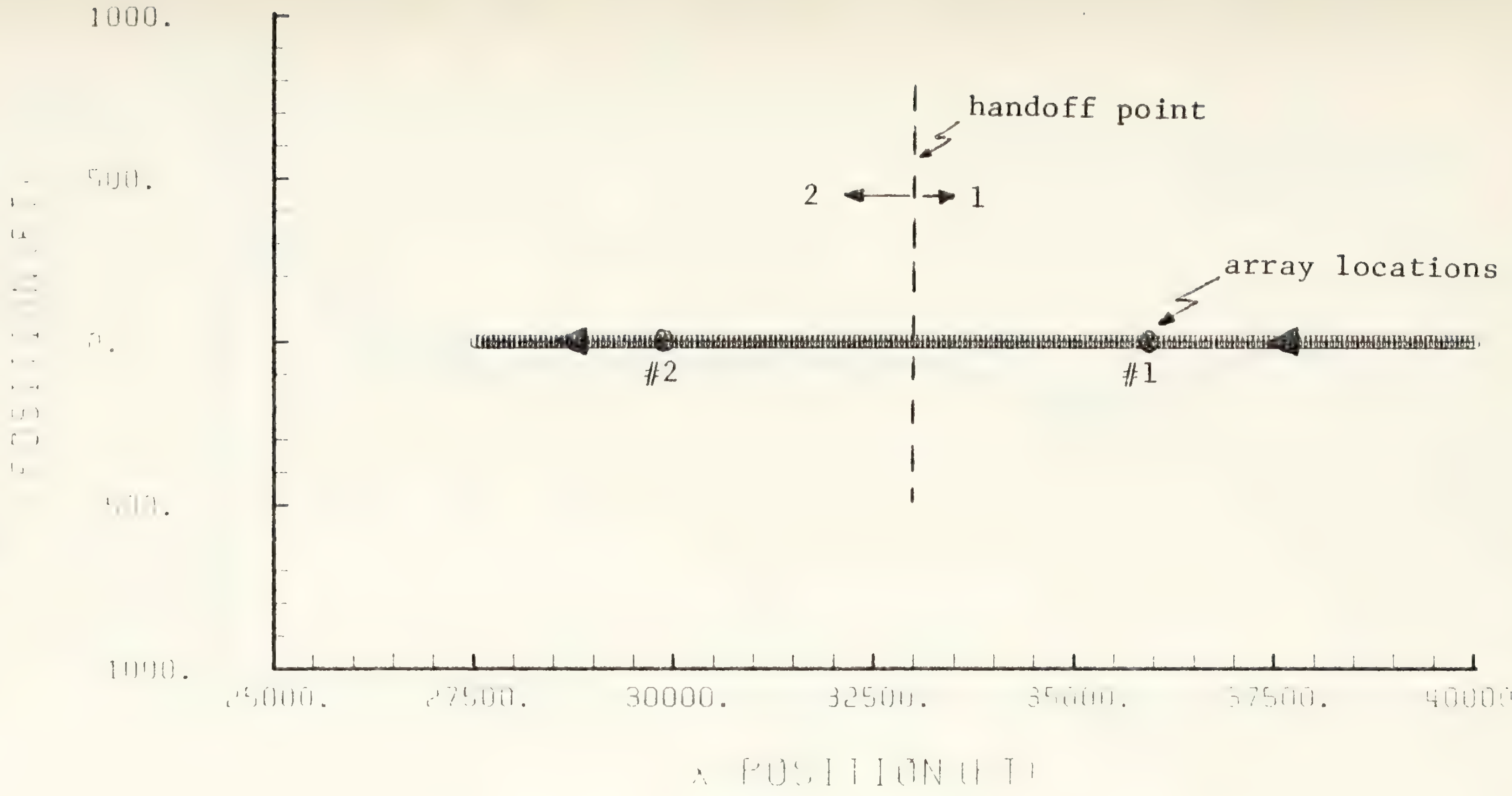


Figure 23. True Trajectory of the Torpedo During a Straight Run Through Multiple Array

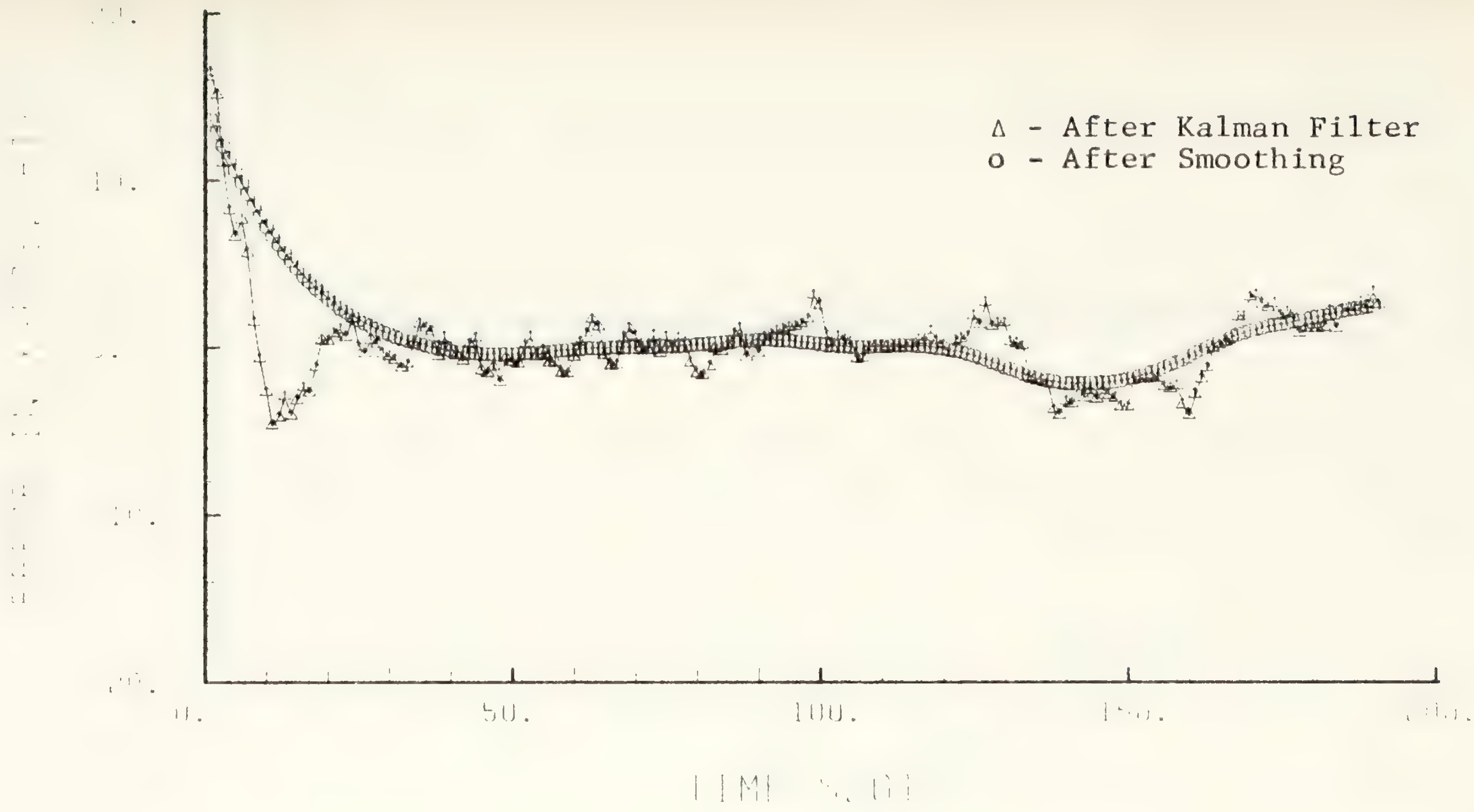


Figure 24. Error in Torpedo X-Position During a Straight Run Through Multiple Array

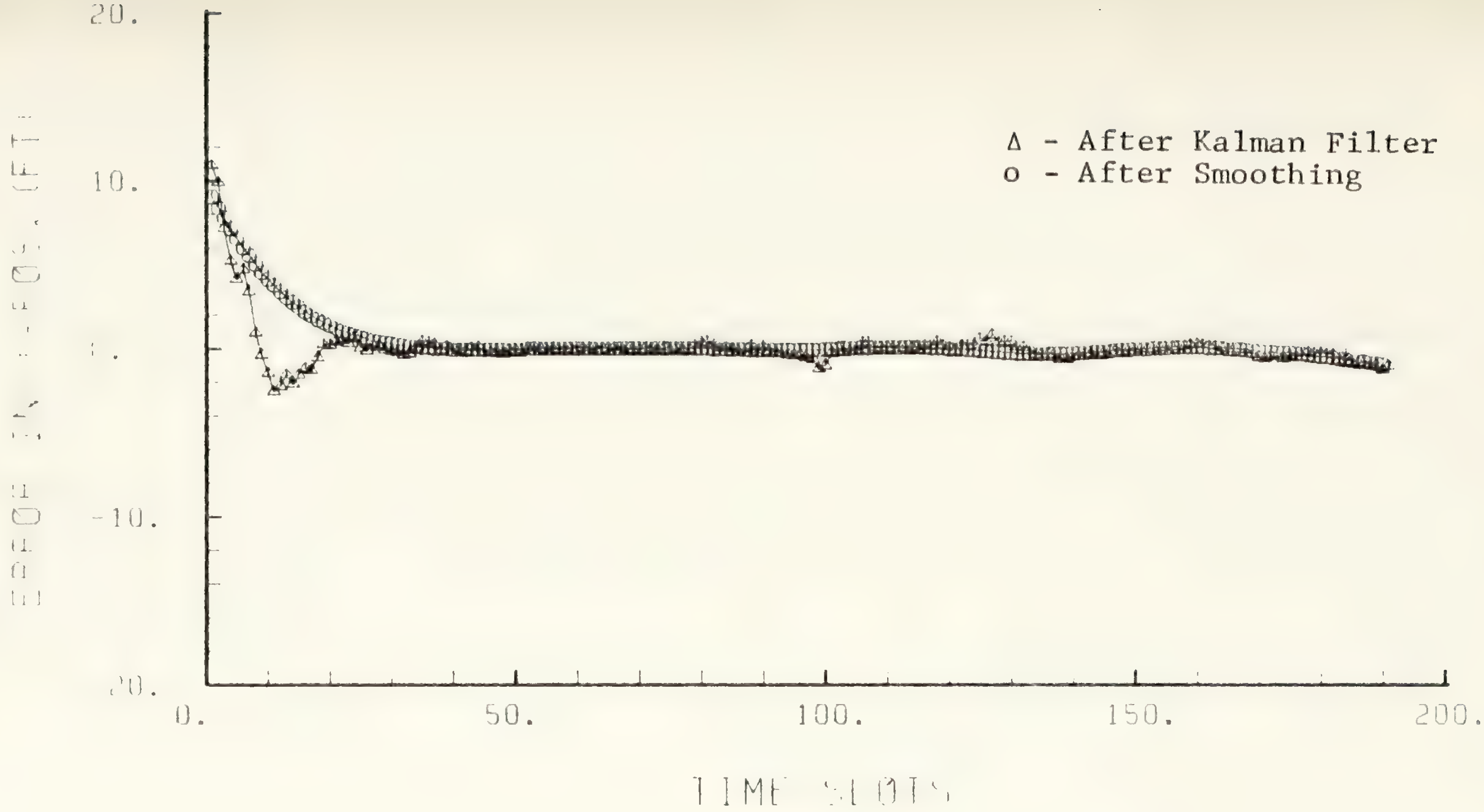


Figure 25. Error in Torpedo Y-Position During a Straight Run Through Multiple Array

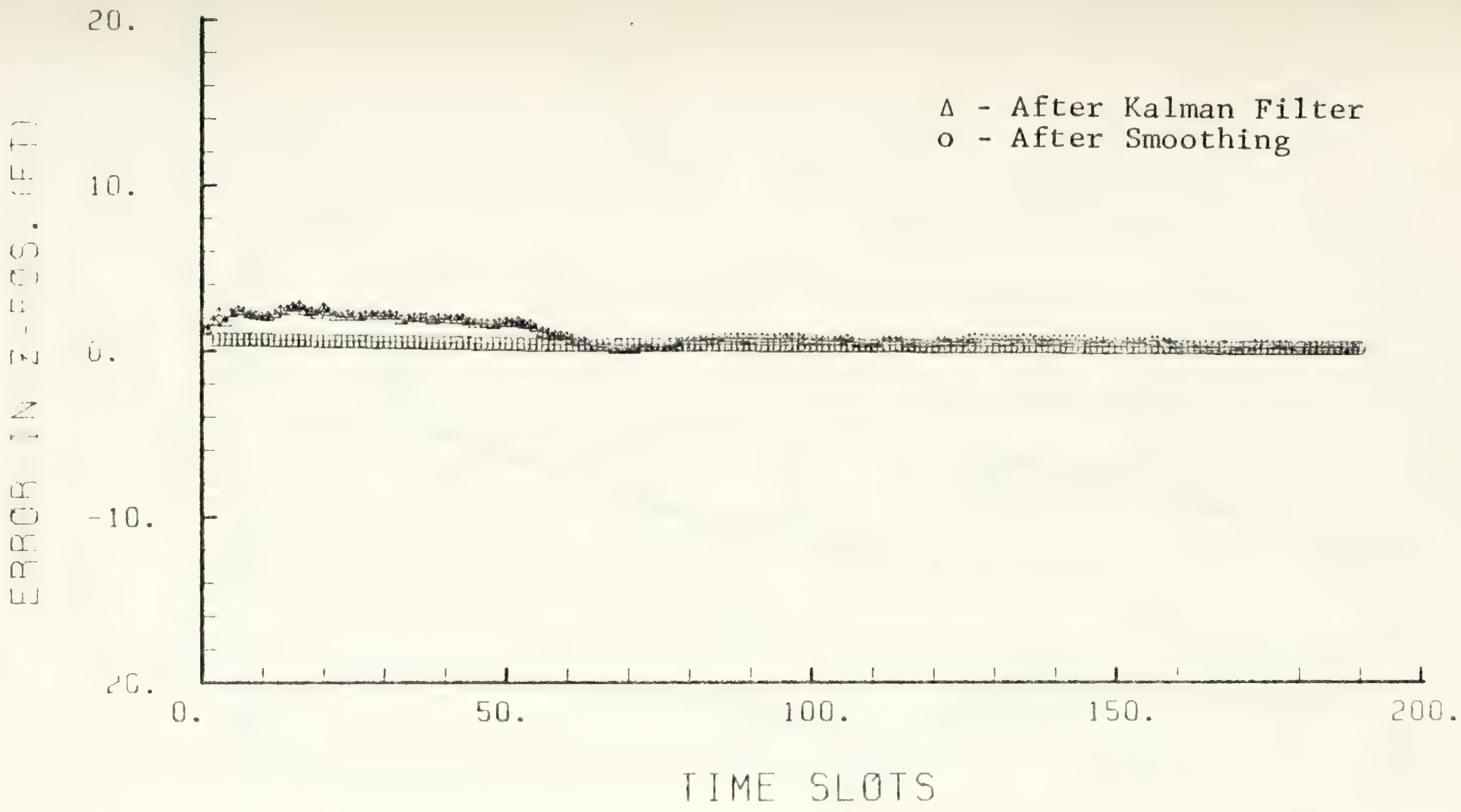


Figure 26. Error in Torpedo Z-Position During a Straight Run Through Multiple Array

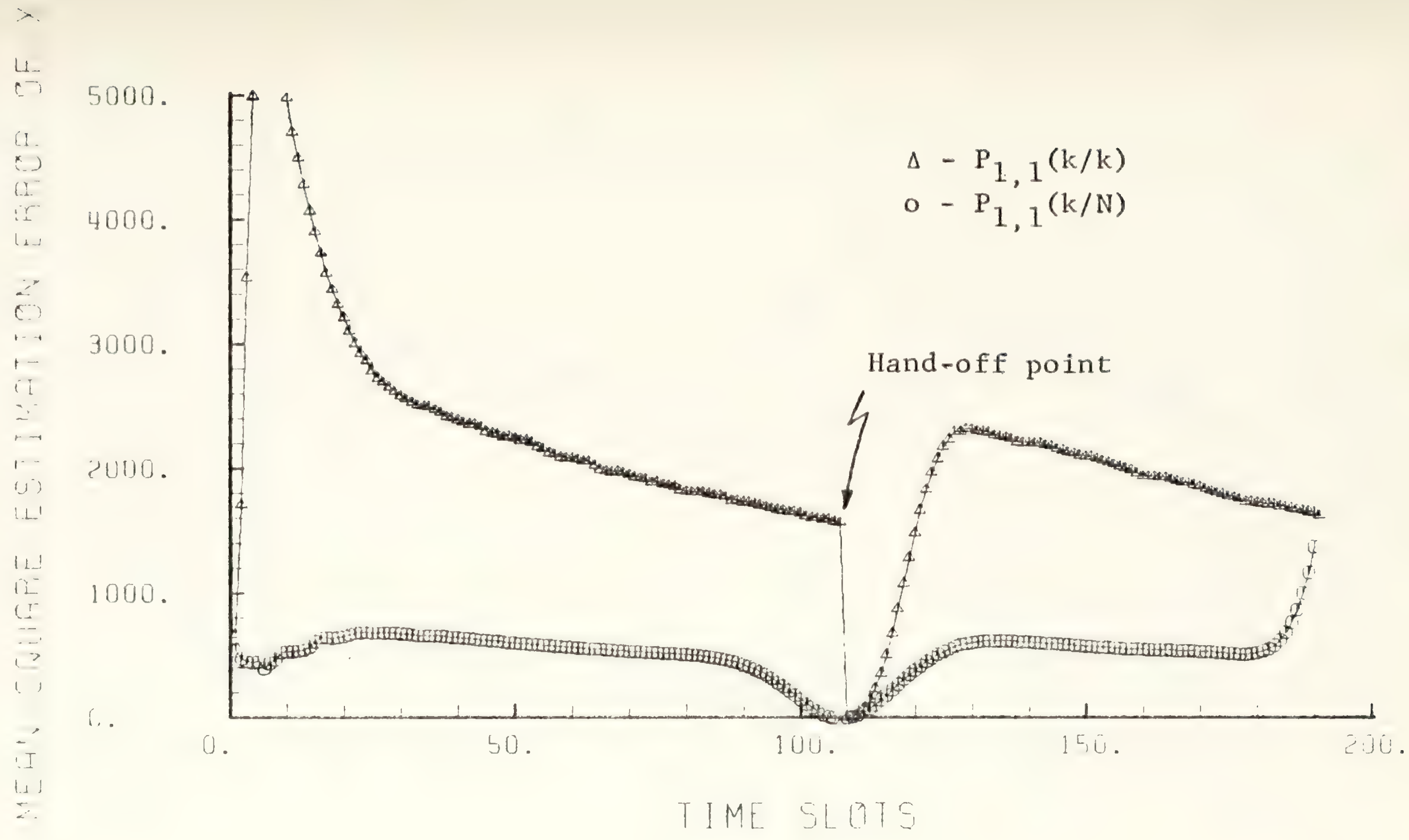


Figure 27. Mean Square Estimation Error (ft.^{2}) in X-Position During a Straight Run Through Multiple Array

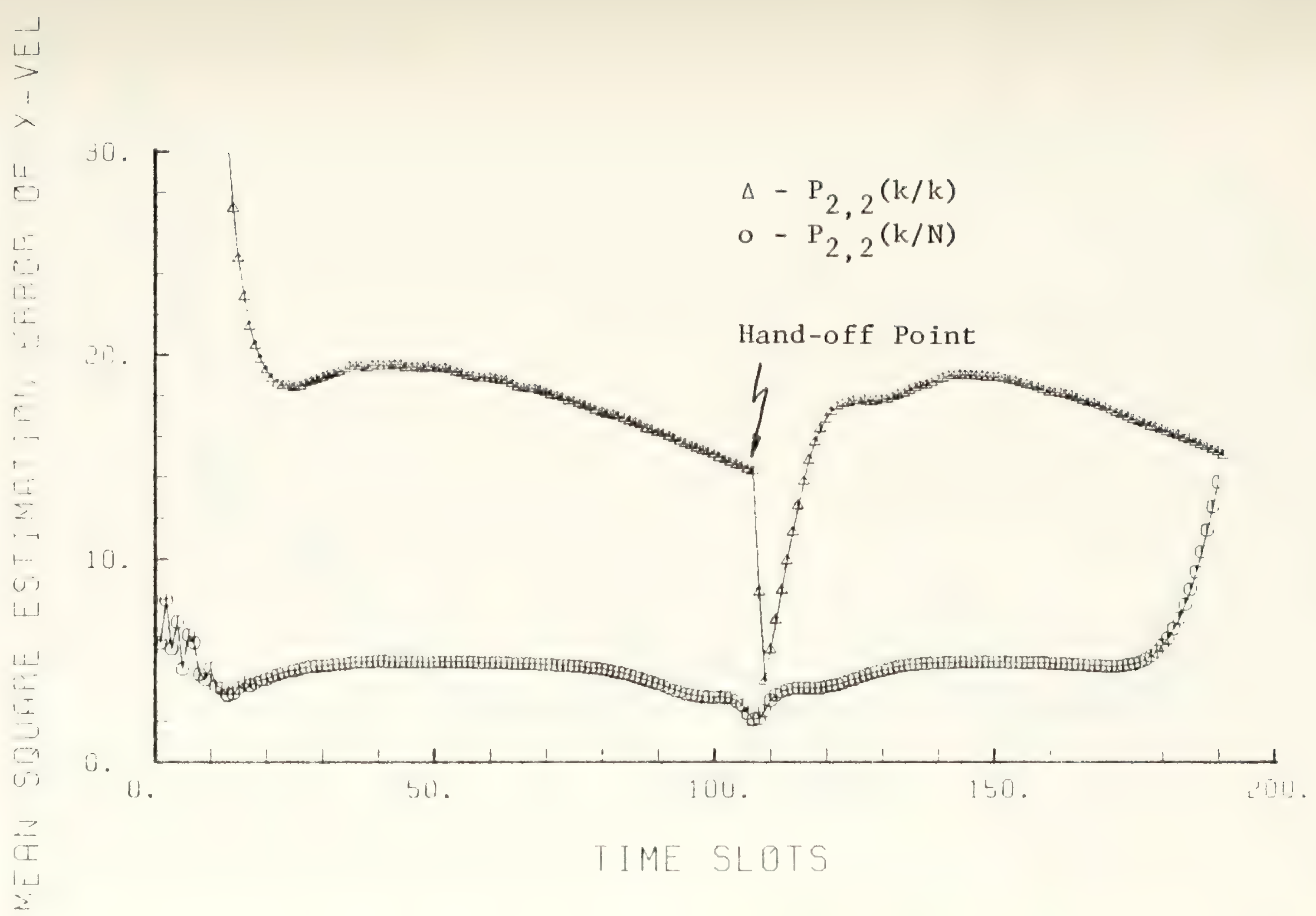


Figure 28. Mean Square Estimation Error ($ft.^2/sec.^2$) in X-Velocity During a Straight Run Through Multiple Array

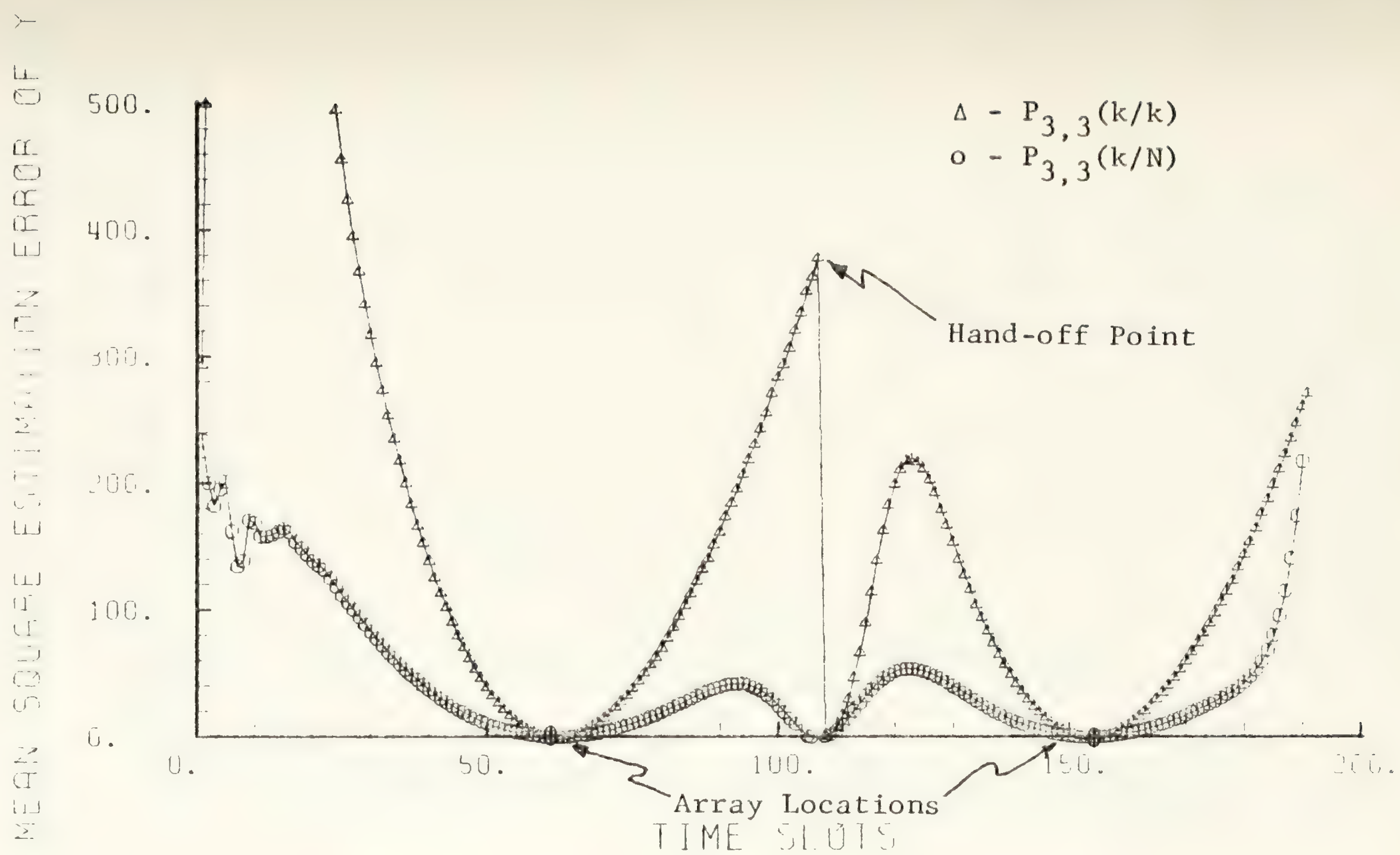


Figure 29. Mean Square Estimation Error (ft.^{2}) in Y-Position During a Straight Run Through Multiple Array

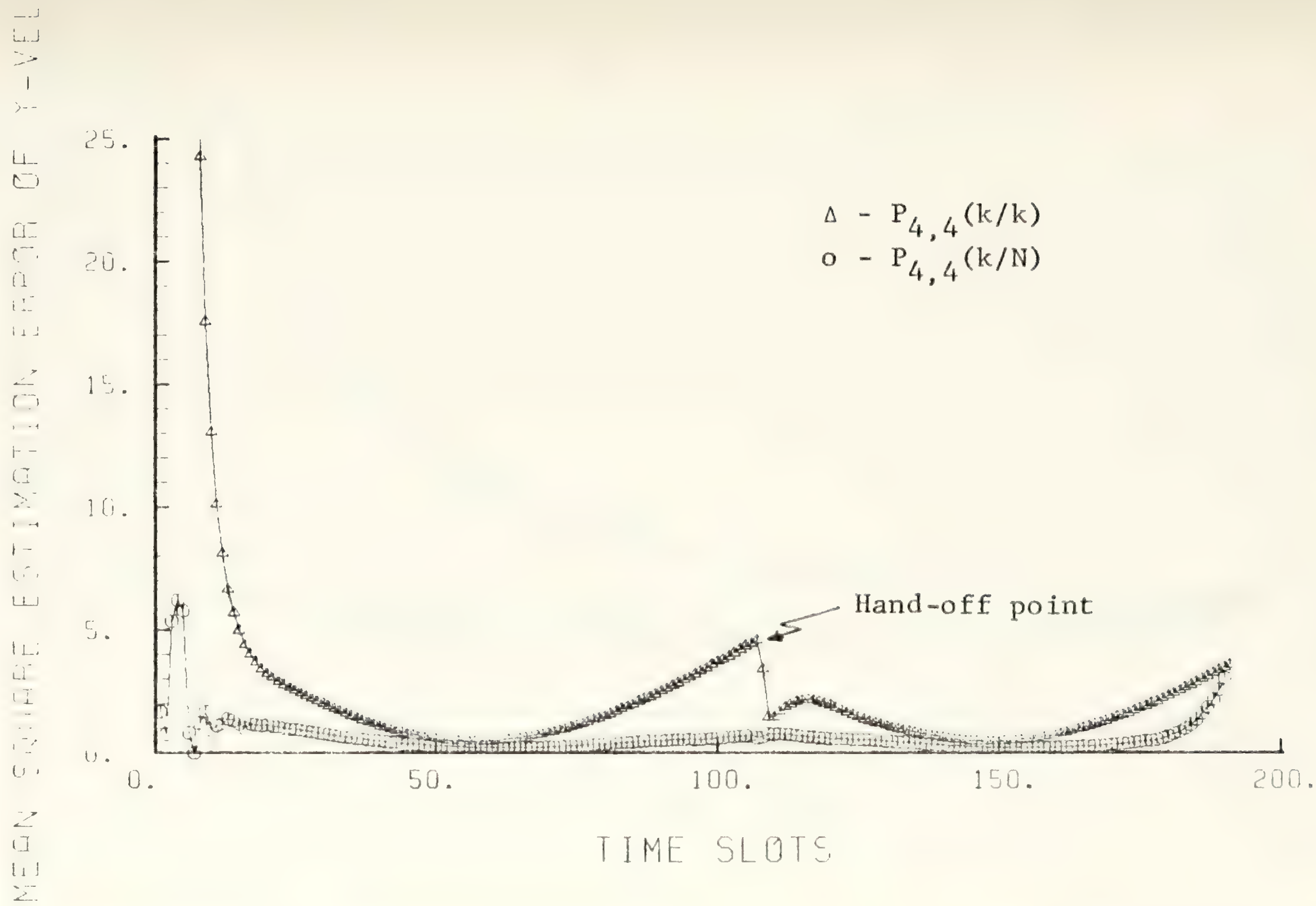


Figure 30. Mean Square Estimation Error (ft.^{2}/sec.^{2}) in Y-Velocity During a Straight Run Through Multiple Array

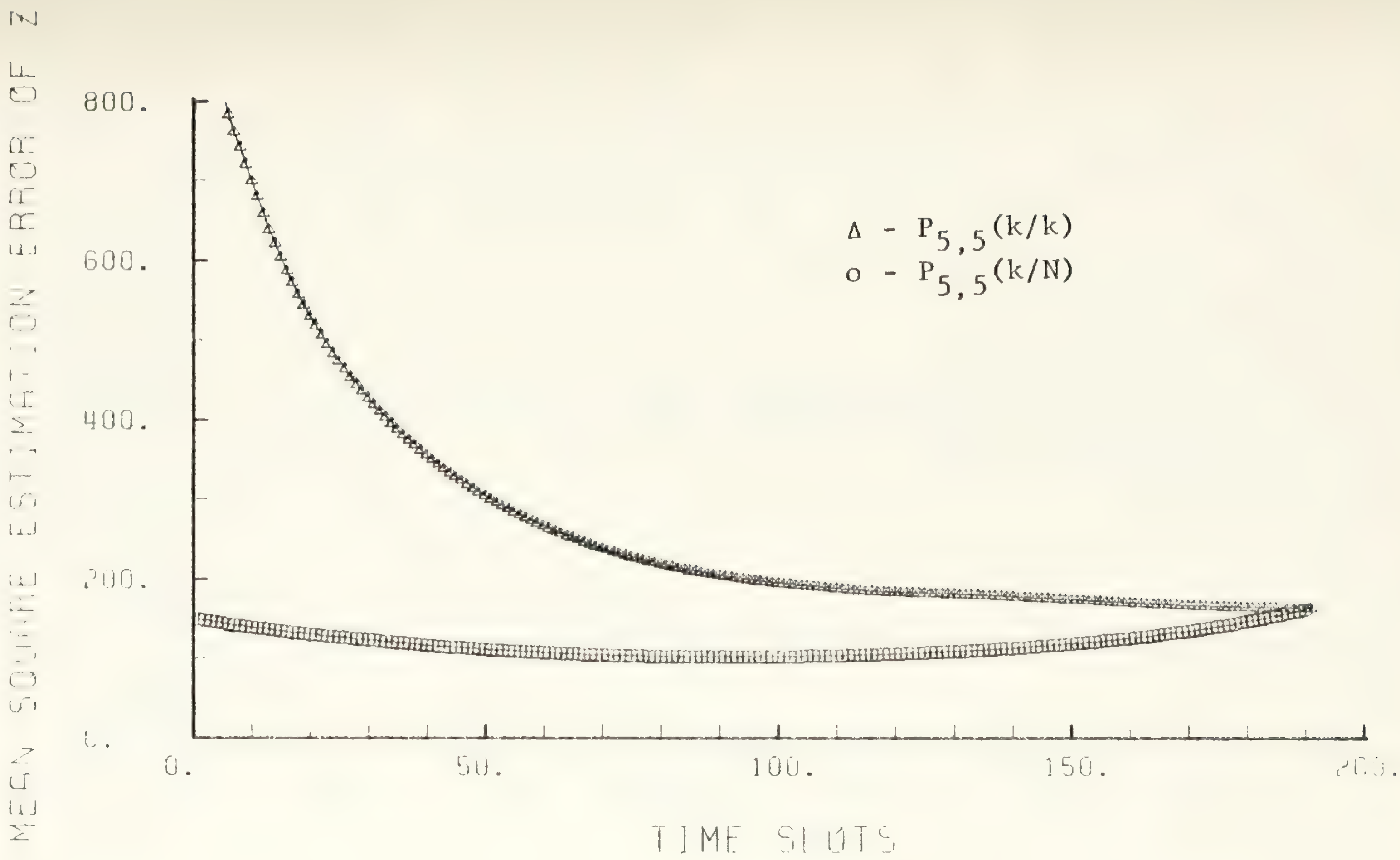


Figure 31. Mean Square Estimation Error (ft.^{2}) in Z-Position During a Straight Run Through Multiple Array

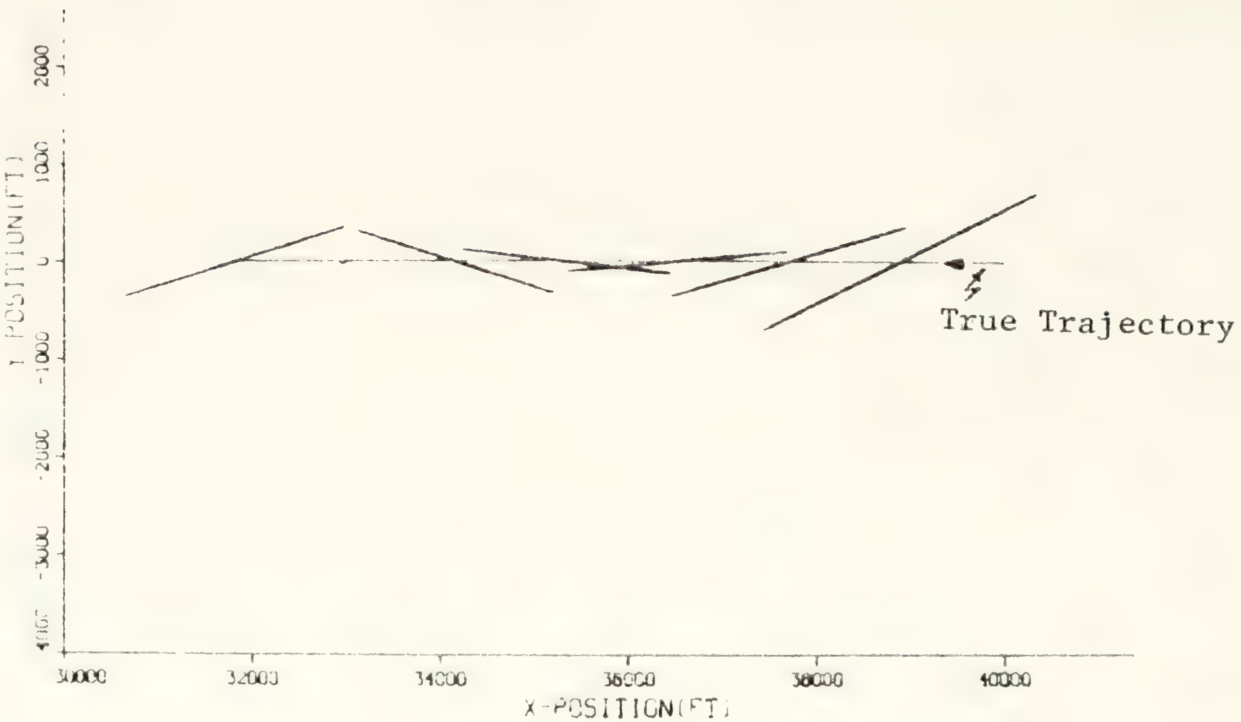


Figure 32. Error Ellipsoids Presented on Every Eighteenth Observation During a Straight Run Through Multiple Array

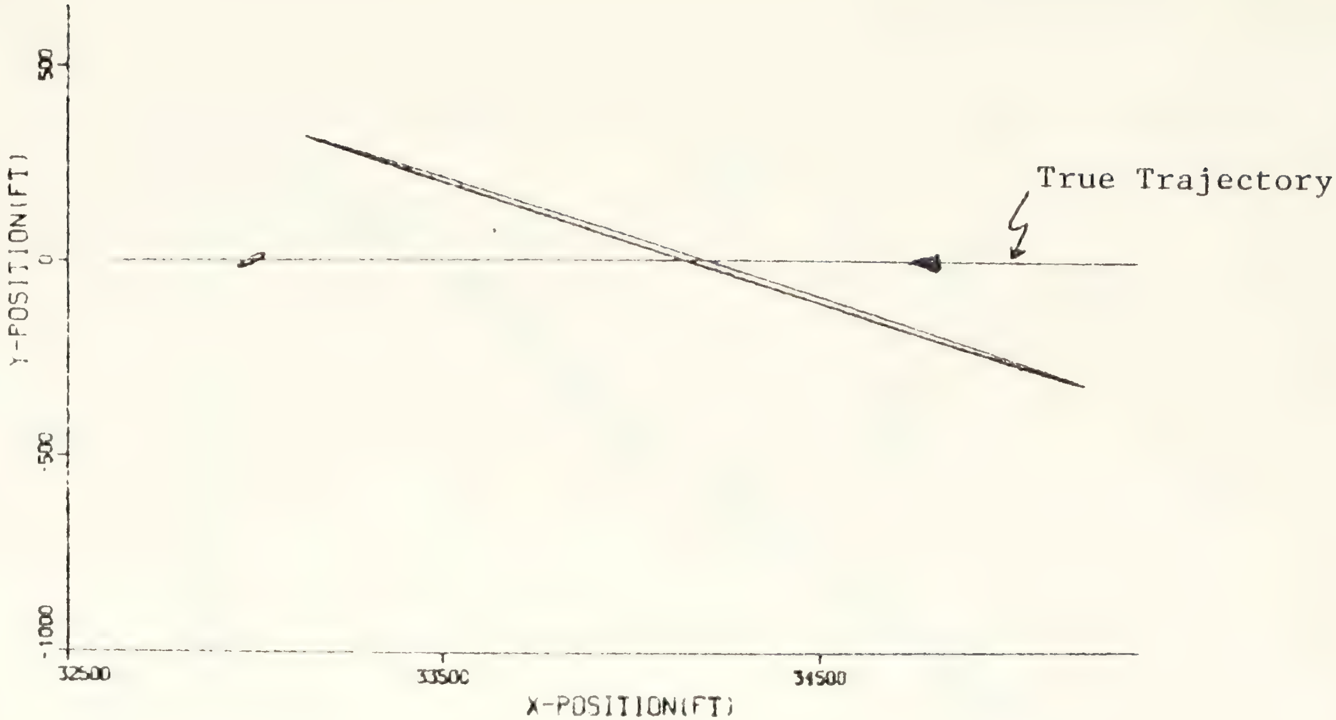


Figure 33. Error Ellipsoids Before and After the Hand-off Point During a Straight Run Through Multiple Array

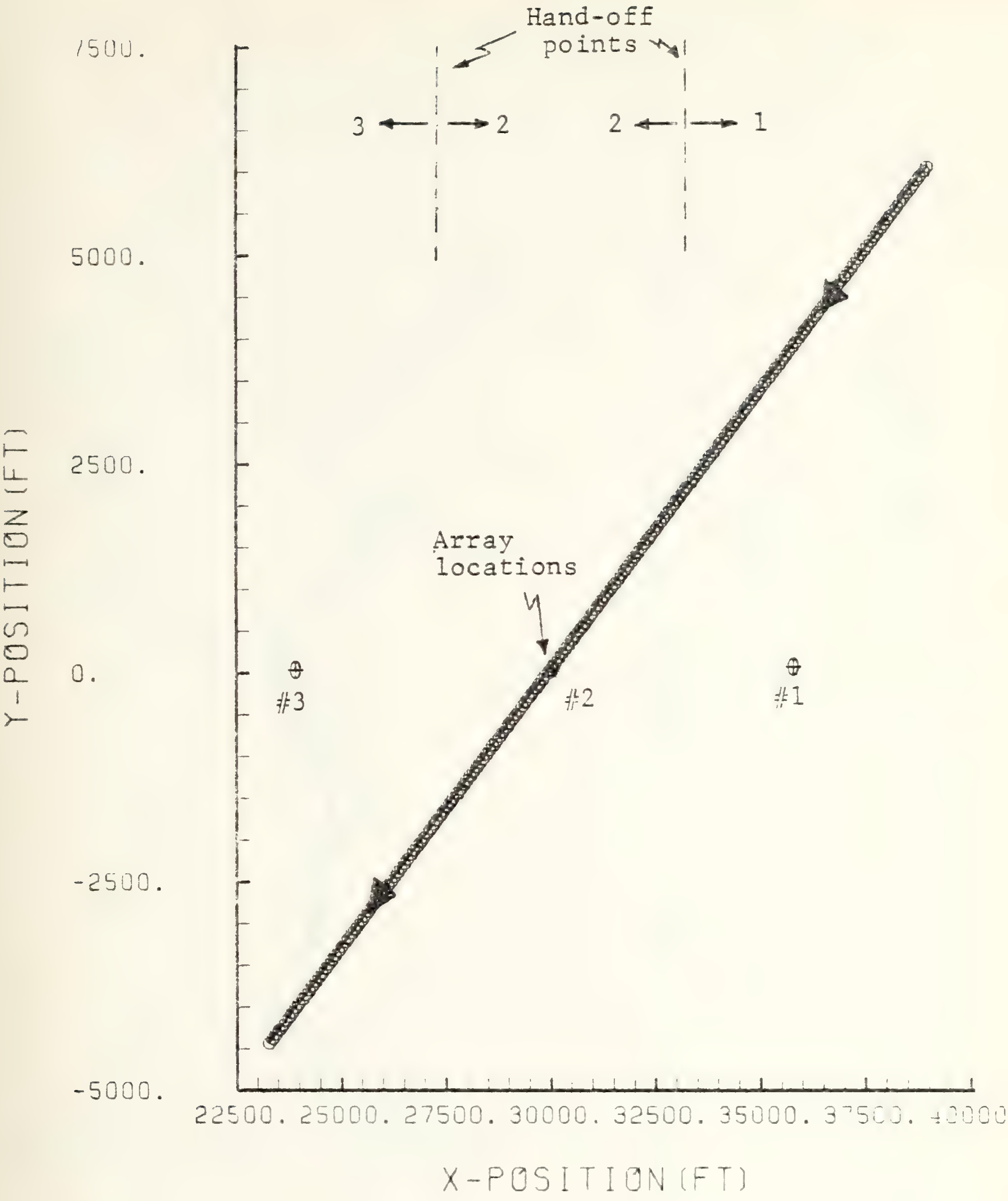


Figure 34. True Trajectory of the Torpedo During a Straight Run Through Multiple Arrays

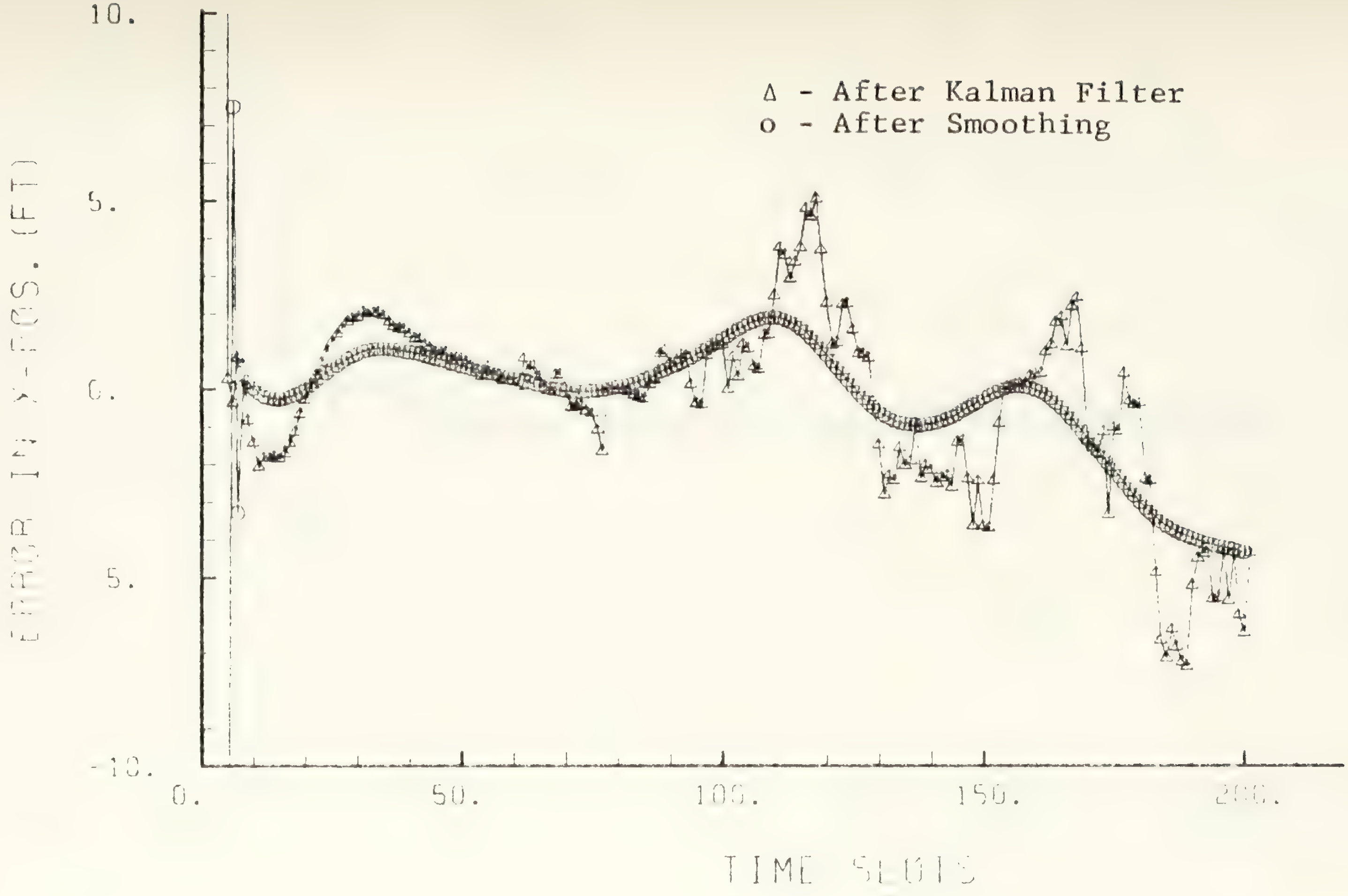


Figure 35. Errors in Torpedo X-Position During a Straight Run Through Multiple Arrays

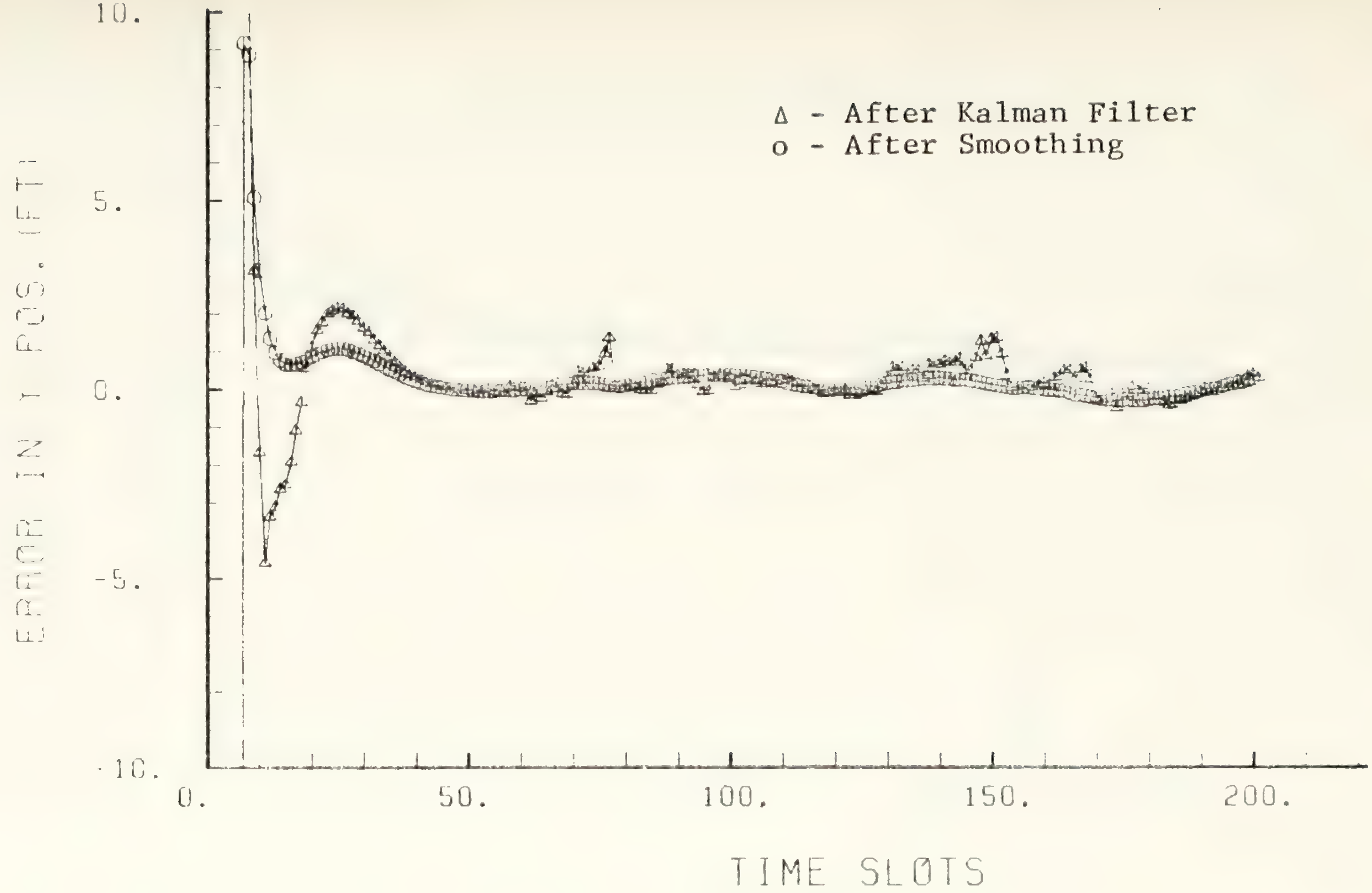


Figure 36. Errors in Torpedo Y-Position During a Straight Run Through Multiple Arrays

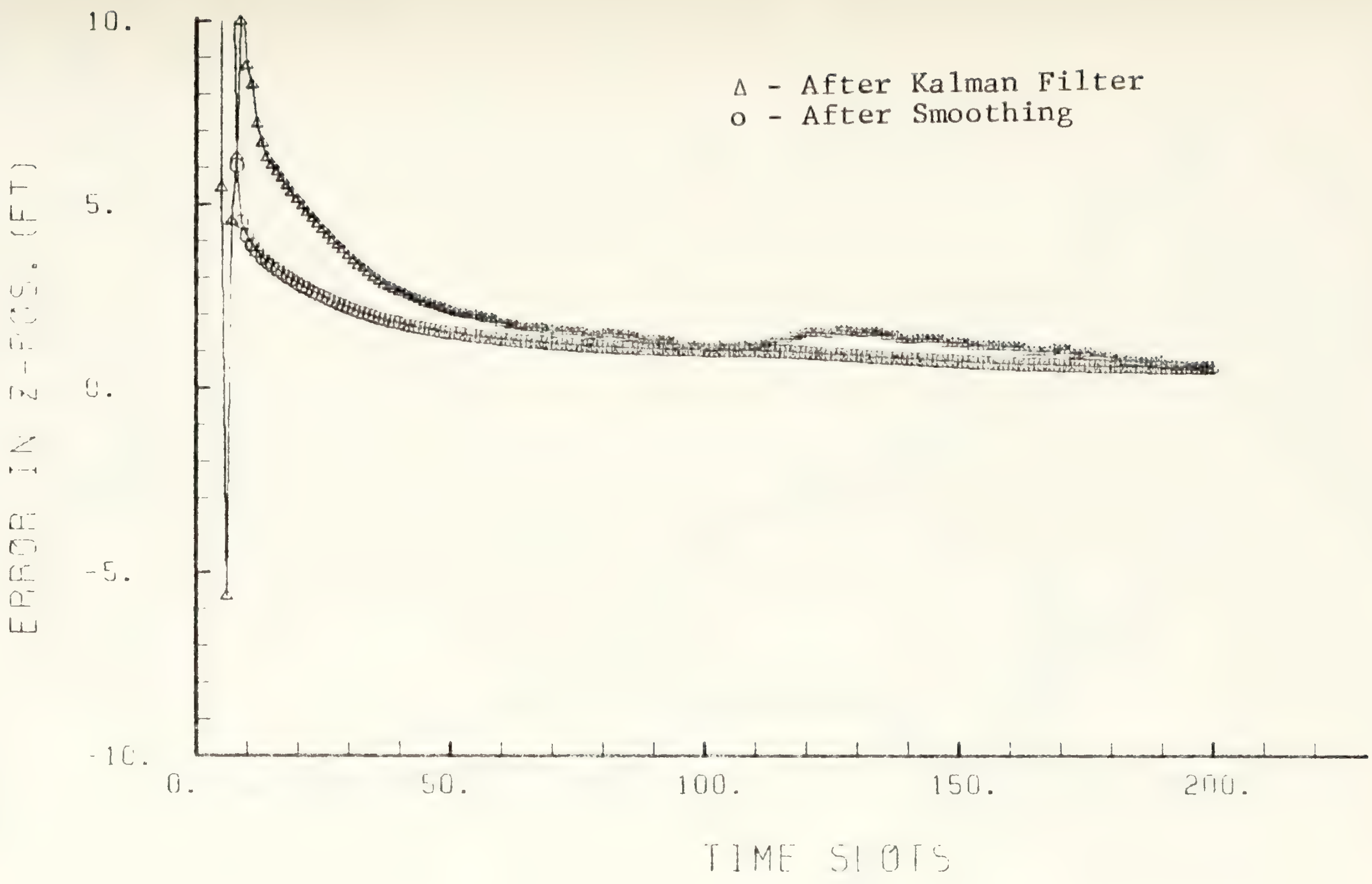


Figure 37. Errors in Torpedo Z-Position During a Straight Run Through Multiple Arrays

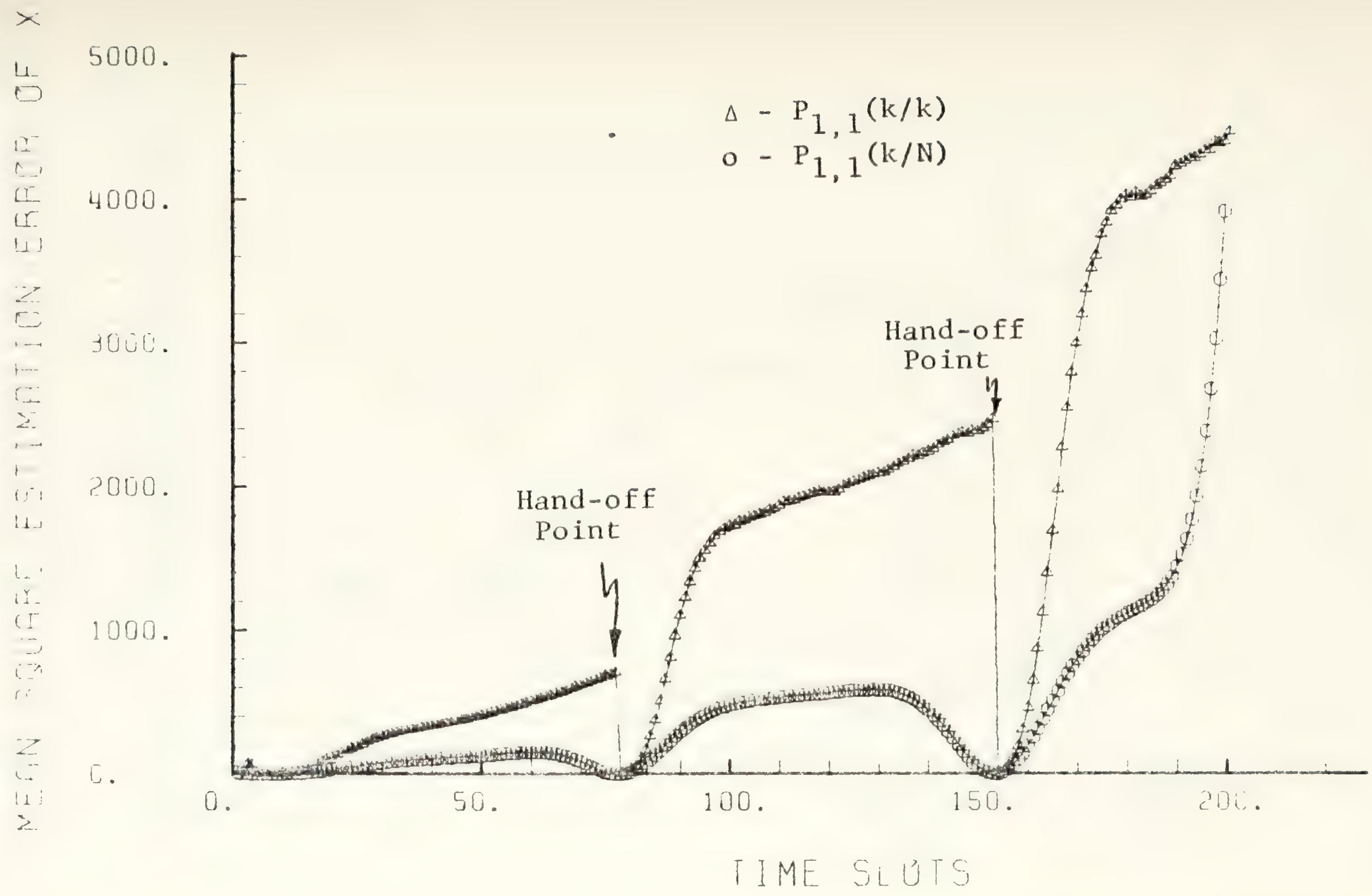


Figure 38. Mean Square Estimation Error (ft.^{2}) in X-Position During a Straight Run Through Multiple Arrays

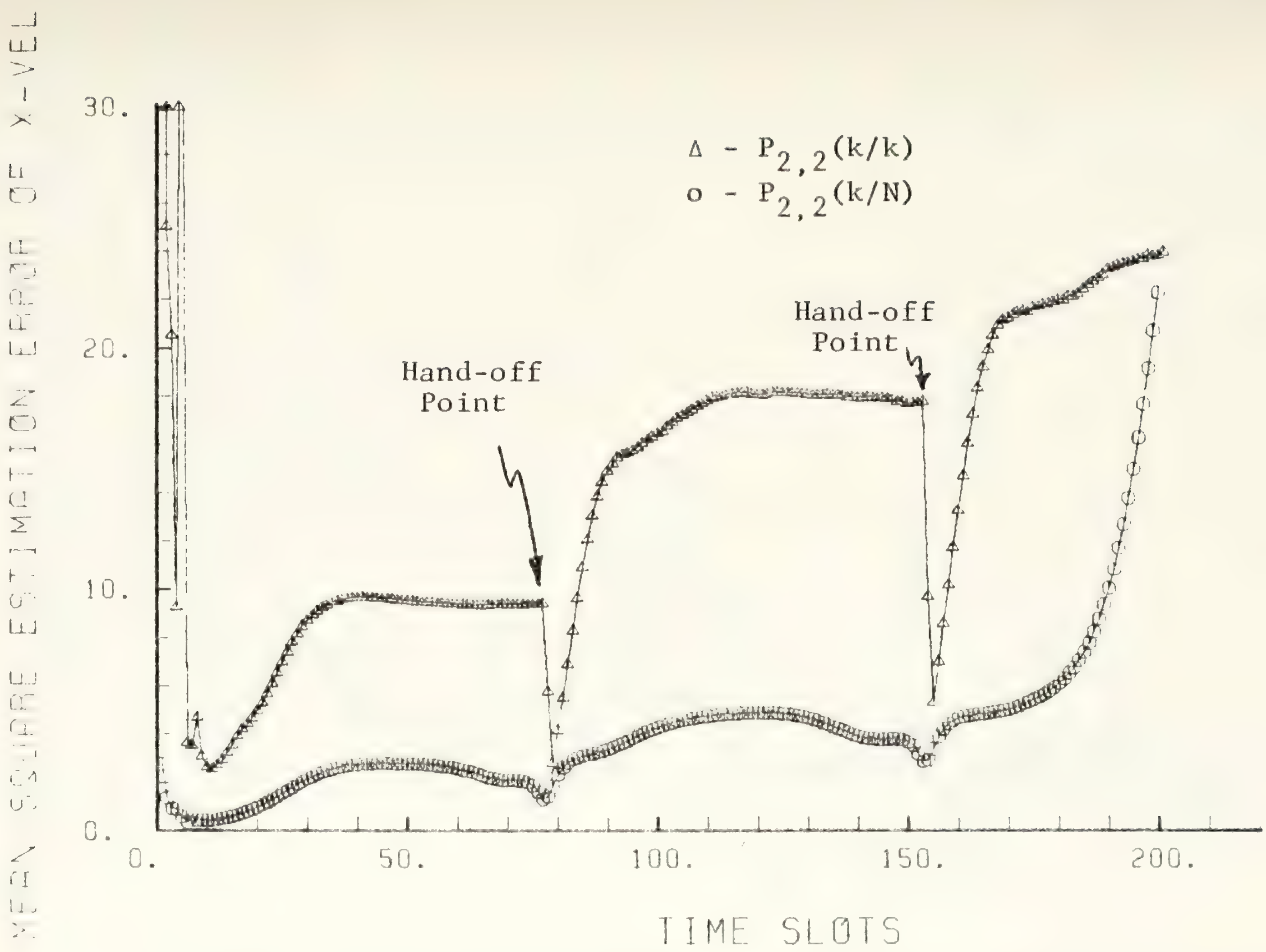


Figure 39. Mean Square Estimation Error (ft.^{2}/sec.^{2}) in X-Velocity During a Straight Run Through Multiple Arrays

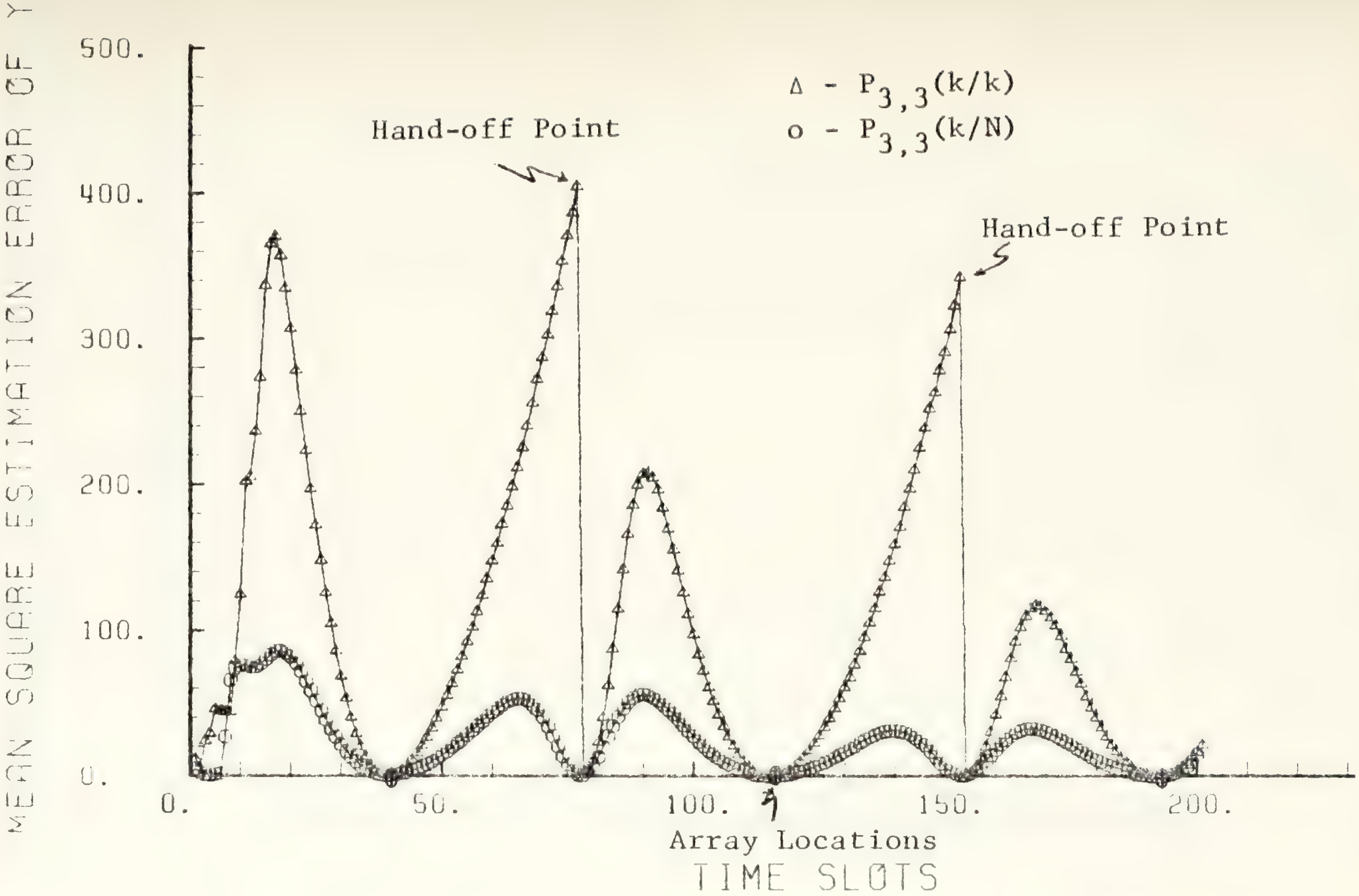


Figure 40. Mean Square Estimation Error (ft.^{2}) in Y-Position During a Straight Run Through Multiple Arrays

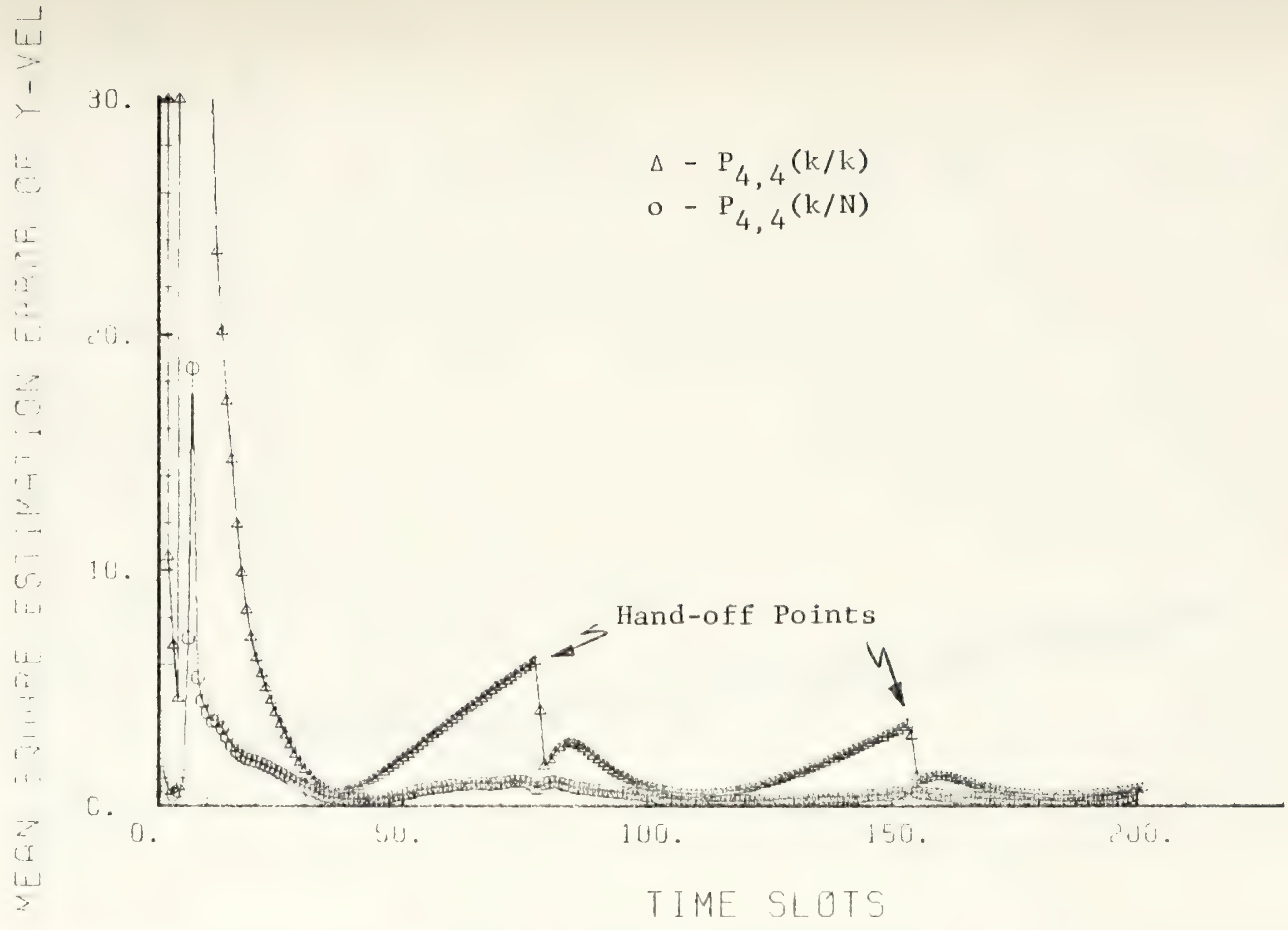


Figure 41. Mean Square Estimation Error (ft.^{2}/sec.^{2}) in Y-Velocity During a Straight Run Through Multiple Arrays

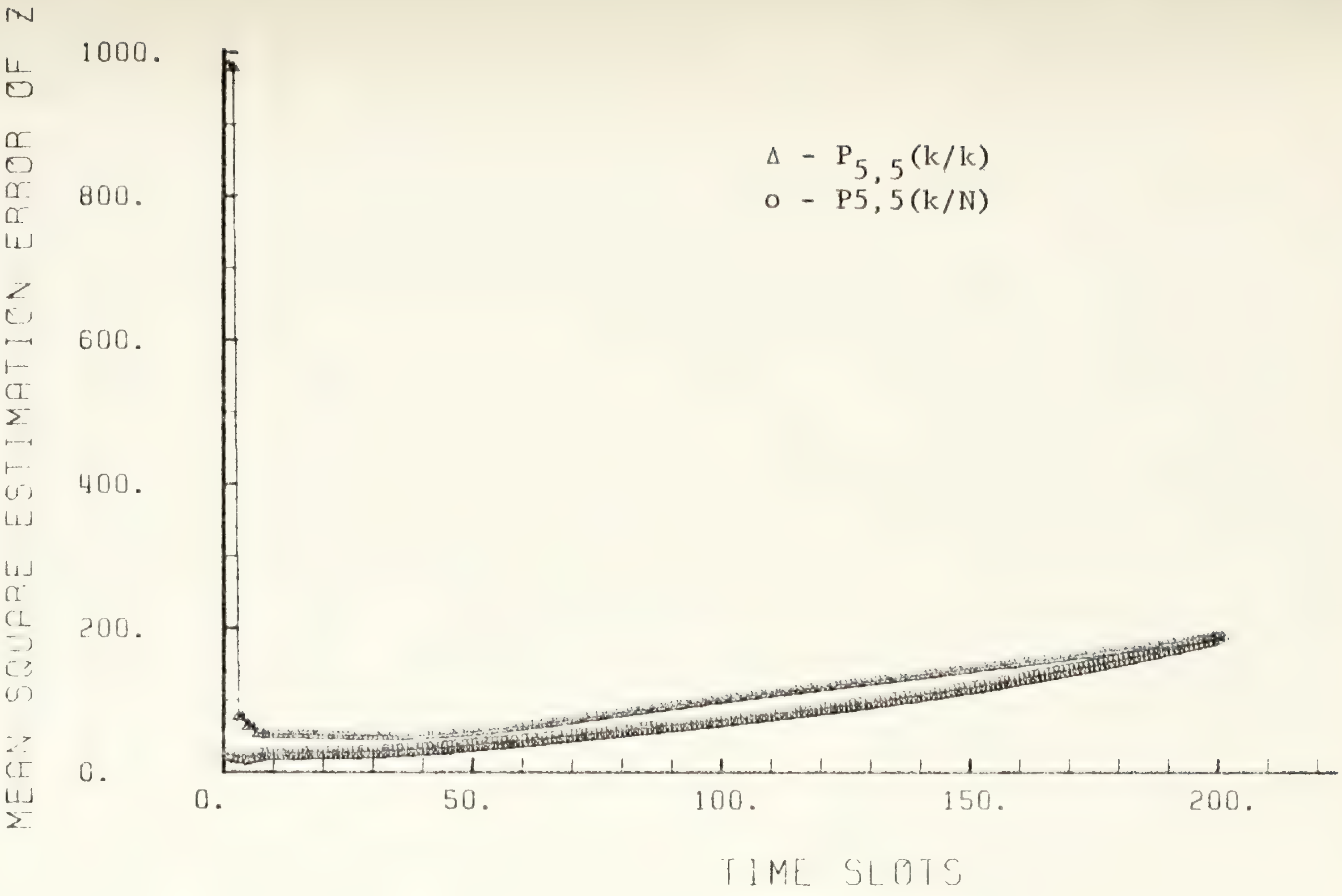


Figure 42. Mean Square Estimation Error (ft.^{2}) in Z-Position During a Straight Run Through Multiple Arrays

APPENDIX A

PROGRAM DESCRIPTION AND FEATURES

The sequential Extended Kalman Filter and Smoothing routine used for torpedo tracking is modularized for ease of implementation. The program is general in nature and many of the parameters of the filter are variable including:

a. The number of states in the filter -- N

b. The number of random forcing functions -- M

c. The number of measurements -- JS

d. Number of time slots -- JTIME

The constant matrices PHI, R, COVW, and GAMMA are initialized in the beginning of the program using data statements. The filter is initialized with P(1/0) and x(1/0) (initial covariance of estimation error and states) using subroutine INIT. The first estimate is at time 1 and continues until ITIME = JTIME + 1. True measurement times (ZI) are computed using either subroutines TRAJEC or TRAJC3, depending on whether single array or multiple array tracking is implemented. Either subroutine will compute four measurement times (T_c, T_x, T_y, T_z) for each time slot. The measurement times are corrupted by zero-mean, white Gaussian noise using the IBM-3033 subroutine GGNML. For each of the four time measurements the corresponding row of the

linearizing H matrix is calculated using either subroutine CHROW or CHROW3, depending on whether single array or multiple array tracking is used. The corresponding gain matrix column GI is then found. These row and column values are utilized in forming the covariance of estimation error. PI, for that particular time measurement. Next the estimate of the observation time ZHAT from that particular hydrophone is formed using the subroutine CZHAT or CZHAT3, depending on whether single or multiple array tracking is implemented. The residual ZDIFF(I) = ZIC(I) - ZHAT, is then calculated. Finally the estimate of the states XI based on one time measurement is calculated and the process is repeated for the next measurement. After four iterations, XI becomes the state estimate XKK and PI becomes the updated covariance of estimation error PKK, and the predictions of the states and covariances XKKM1 and PKKM1 are formed. Finally, for each time slot (except the first) smoothed state estimates, XKKS, and covariances, PKKS, are formed using the subroutine SMOOTH. PLOTP is used to generate line printer plots and PLOTG is used to generate VERSATEC plots.

A. PROGRAM SUBROUTINES

A brief description of the subroutines are described below:

1. TRAJEC -- This subroutine develops the torpedo trajectory which is used as truth data for the filter. The

subroutine outputs true transit times, ZI(I), and the x, y, z positions, TD(I), of the torpedo for each time slot. THe subroutine is used during single array tracking.

2. TRAJC3 -- This subroutine performs the same function as TRAJEC but is used only during multiple array tracking.

3. INIT -- This subroutine generates the initial state vector x(0/-1) and initial covariance matrix P(0/-1).

4. CHROW -- This subroutine computes the appropriate row of the linearizing H matrix. Each row corresponds to one of the four transit time measurements, T_c, T_x, T_y, T_z. This subroutine is used during single array tracking.

5. CHROW3 -- This subroutine performs the same functions as CHROW but is used only during multiple array tracking.

6. CZHAT -- This subroutine computes the estimated transit times for the filter. Four transit times, ZHAT, are calculated corresponding to each of the four true transit times ZI(I). This subroutine is used during single array tracking.

7. CZHAT3 -- Subroutine performs same functions as CZHAT however it is used only during multiple array tracking.

8. QFIND -- This subroutine develops an adaptive Q matrix which is a function of the torpedo velocity. Three input variables defined in a data statement at the beginning of the program can be adjusted:

aa. SIGACC -- Maximum expected horizontal acceleration of the torpedo.

bb. SIGDIV -- Maximum expected change in vertical velocity.

cc. SIGCC -- Maximum expected turn rate of the torpedo in the horizontal plane.

The values listed in the program were used and kept constant during the simulation tests. If the user desires not to use the adaptive Q subroutine, software code is provided at the beginning of the program to calculate a constant Q matrix.

9. GGNML -- This is an IBM-3033 subroutine contained in the IMSL library. The routine generates zero mean white Gaussian noise with an RMS value normalized to 1. The main program scales the noise and adds it to the transit time measurements.

10. PLOTP -- This is an IBM-3033 subroutine used to generate the line printer plots. Information on this subroutine can be obtained from the IMSL library.

11. PLOTG -- This is an IBM-3033 subroutine used to generate the VERSATEC plots. Information on this subroutine can be obtained from the IMSL library.

12. SMOOTH -- This subroutine computes the smoothed state estimates and covariances.

B. UTILITY PROGRAMS

These subroutines were designed to be used for repetitive matrix and vector manipulations:

1. PROD -- multiplying two matrices
2. MMULT -- multiplying a matrix and a vector
3. VMULT -- multiplying two vectors
4. TRANS -- transposing a matrix
5. ADD -- adding two matrices
6. SUB -- subtracting two matrices
7. RECIP -- inversing a matrix

APPENDIX B

SEQUENTIAL EXTENDED KALMAN FILTER AND SMOOTHING PROGRAM LISTING

```
C     DOUBLE PRECISION XKKM1,PKKM1,PKK,XKK,PHI,GAMMA,R,PHIPKK,PHIT,
C    1PKTEMP,HROW,PI,PDUM,QTEMP,GNUM,GI,XI,ZI,GAMMAT,Q,COVW,ZC,ZIC,
C    2ZCS,ZHAT,DATR,ZDIFF,GDENOM,GDTEMP,ZDIFAV
      REAL*4     PHI(5,5),GAMMA(5,3),R(4,4),PHIPKK(5,5),PHIT(5,5),
     1PKTEMP(5,5),HROW(5),PI(5,5),PDUM(5,5),QTEMP(5,3),ZDIFF(4),
     2GNUM(5),GI(5),XI(5),ZI(4),GAMMAT(3,5),Q(5,5),COVW(3,3),ZIC(4),
     3ZCS,ZHAT,GDENOM,GDTEMT,ZDIFAC,XTRK(5),SS(5,210),
     4SS1(210,5,5),SQ1(210,5,5),SIGCC,SIGACC,SIGDIV
      REAL*4  KOUNT(210),AKOUNT(210),PVEC(210),X3(210),XDIX(210),
     1TRUX(210),TRUY(210),TRUZ(210),TD(6),C1(2),C2(2),C3(2),X1(210),
     2C4(2),C5(2),C6(2),C7(2),XDIFF1(210),XDIFF3(210),X9(210),
     3XDIFF5(210),HYDRO(6,12),XB(4),YB(4),ZB(4),DATR(17),
     4GX(210,4),GY(210,4),ZDIFF1(210,4),ZDM(2),GXM(2),GYM(2),X5(210),
     5P3P3(210),XP4S(210),XP4K(211),
     6P4P4(211),XP5S(211),XP5K(211),
     7P5P5(211),XP3S(210),XP3K(211),ZKER(211),ZSER(211),P5(210,5,5),
     8P2P2(210),XP2S(210),XP2K(211),YKER(211),YSER(210),YP88(7,25),
     9P1P1(211),XP1S(210),XP1K(211),XKER(211),XSER(210),XP88(7,25),
     0P1PP(210),P2PP(210),P3PP(210),P4PP(210),P5PP(210),SY(7),
     1P1K(210),P2K(211),P3K(211),P4K(211),P5K(211),XP6(5,210),SX(7),
     2P1P(211),P2P(211),P3P(211),P4P(211),P5P(211),RN(4),SIGY(7),
     3XP7(210),XP8(210),XP9(210),XP10(210),XP11(210),XP12(210),SIG1(7),
     4XP13(210),XP14(210),XP15(210),XP19(25),YP19(25),ANGL(7),ANGLD(7),
     5XP29(25),YP29(25),XP39(25),YP39(25),XP49(25),YP49(25),SIG2(7),
     6XP59(25),YP59(25),XP69(25),YP69(25),XP79(25),YP79(25),SIGX(7),
C
      COMMON/THE1/XKKM1(5)/THE3/XP(5,210)/THE4/P1(210,5,5)/THE5/SI
      COMMON/THE2/PKKM1(5,5),PKK(5,5),XKK(5)
C-----------------------------------------------------------------------
C     INITIALIZE CONSTANTS,HYDROPHONE MATRIX,PHI,R,COVW,GAMMA,
C     AND DATA FOR TRAJECTORY
C-----------------------------------------------------------------------
      DATA N/5/,M/3/,JS/4/,NZDIFF/4/,JTIME/45/,
C    1 SIGACC/0.0/,SIGDIV/0.0/,SIGCC/0.0/,
     2 HYDRO/36000.,30000.,24000.,18000.,12000.,6000.,6*6000.,
     3    6*0.0,36030.,30030.,24030.,18030.,12030.,6030.,6*6000.,
     4    6*0.0,36000.,30000.,24000.,18000.,12000.,6000.,6*6030.,
     5    6*0.0,36000.,30000.,24000.,18000.,12000.,7*6000.,6*30./
C
       DATA PHI/1.0,4*0.0,1.31,1.0,5*0.0,1.0,4*0.0,1.31,1.0,5*0.0,1.0/,
     2 R/1.0E-8,4*0.0,1.0E-8,4*0.0,1.0E-8,4*0.0,1.0E-8/,
     3COVW/1.,3*0.0,1.,3*0.0,1./,
     4 GAMMA/.86,1.31,5*0.0,.86,1.31,5*0.0,1.31/,
C-----------------------------------------------------------------------
C     DATA STATEMENT DEFINED IN SUBROUTINE TRAJEC OR TRAJC3
C-----------------------------------------------------------------------
     5DATR/7500.,1300.,0.,-40.,0.,3*0.0,3*0.0,4.71,.174,.1,8.1,
     6600.,800./
```

```
C------------------------------------------------------------
C      THE NEXT THREE STATEMENTS ARE REQUIRED FOR
C      SUBROUTINE WHICH GENERATES WHITE NOISE
C------------------------------------------------------------
       DOUBLE PRECISION   DSEED
       NR=4
       DSEED=76869.D0
       SIGCC=SIGCC*3.14159/180.
C------------------------------
C      LOAD X(0/-1),P(0/-1)
C------------------------------
       CALL INIT(XKKM1,PKKM1)
C------------------------------
C      GET TRANSPOSES
C------------------------------
       CALL TRANS(GAMMA,N,M,GAMMAT)
       CALL TRANS(PHI,N,N,PHIT)
       WRITE(6,163)
C
 163    FORMAT('0','R MATRIX')
       DO 264 I=1,4
 264   WRITE(6,136)(R(I,J),J=1,4)
 136    FORMAT(4F14.11)
       WRITE(6,265)
 265    FORMAT('0','COV W MATRIX')
       DO 266 I=1,3
 266   WRITE(6,267)(COVW(I,J),J=1,3)
 267   FORMAT(3F11.6)
C------------------------------
C CALCULATE THE Q MATRIX
C------------------------------
       CALL PROD(GAMMA,COVW,N,M,M,QTEMP)
       CALL PROD(QTEMP,GAMMAT,N,M,N,Q)
       WRITE(6,100)
 100   FORMAT('0','Q MATRIX')
       DO 101 LI=1,N
 101   WRITE(6,102)(Q(LI,LJ),LJ=1,N)
 102   FORMAT(1X,6F15.4)
C------------------------------------------------------------
C START THE TIME SLOT LOOP AND SET ARRAY HANDOFF POINT
C------------------------------------------------------------
       ITIME=JTIME+1
       I8=1
       XT=36000.
       SW=XT-3000.
         DO 99 KK=1,ITIME
            WRITE(6,6) KK
 6          FORMAT (//1X,100('*'),'TIME=',I4//)
```

```
C-----------------------------------------------------------
C      GET HYDROPHONE ARRAY COORDINATES
C-----------------------------------------------------------
 601   DO 600 I3=1,4
       I4=3*I3
       I5=I4-2
       I6=I4-1
       XB(I3)=HYDRO(I8,I5)
       YB(I3)=HYDRO(I8,I6)
 600   ZB(I3)=HYDRO(I8,I4)
       WRITE(6,602)
 602   FORMAT(' ',' HYDROPHONE COORDINATES')
       WRITE(6,*)XB
       WRITE(6,*)YB
       WRITE(6,*)ZB
       IF(XKKM1(1).GT.SW)GO TO 610
       I8=I8+1
       WRITE(6,759)I8,KK
 759   FORMAT ('0',15('*'),'ARRAY',1X,I2,'STARTS TRACKING AT TIME',1X,I3,
     1 15('*')/)
       SW=SW-6000.
       XT=XT-6000.
       GO TO 601
       WRITE(6,67)
 67    FORMAT('0','XKKM1')
       DO 51 LA=1,N
 51    WRITE(6,50) XKKM1(LA)
 50    FORMAT(1X,5F14.4)
C-------------------------------------------------------------
C COMPUTE THE TRUE TIMES AND TRUE POSITIONS
C-------------------------------------------------------------
C
 610   CALL TRAJC3(DATR,ZI,TD,XB,YB,ZB)
C610   CALL TRAJEC(KK,DATR,ZI,TD)
       WRITE(6,700) (ZI(J),J=1,4)
 700   FORMAT(' TRUE TIMES : ',4F14.5)
       WRITE(6,701) (TD(J),J=1,3)
 701   FORMAT(' TRUE POSITIONS : ',3F11.4)
       T2=(.25)*(ZI(1)+ZI(2)+ZI(3)+ZI(4))
       IF(KK.EQ.1)T1=T2
       PHI(3,4)=PHI(1,2)
       PHIT(2,1)=PHI(1,2)
       PHIT(4,3)=PHI(1,2)
       T1=T2
       A14=PHI(1,2)
          TRUX(KK)=TD(1)
          TRUY(KK)=TD(2)
          TRUZ(KK)=TD(3)
```

```
C
      DO 52 LA=1,N
 52   WRITE(6,50) XKKM1(LA)
C-----------------------------------------------------------------------
C        FIRST GET HROW-CALCULATE GAIN,ESTIMATE,COVARIANCE OF
C        ERROR BASED ON ONE TIME MEASUREMENT-TC,TX,TY, TZ
C-----------------------------------------------------------------------
 711        DO 97 I=1,JS
     NZDIFF=4
        CALL CHROW3(I,HROW,XB,YB,ZB)
C       CALL CHROW(I,HROW)
         CALL MMULT(PKKM1,HROW,N,N,GNUM)
         CALL VMULT(HROW,GNUM,N,GDTEMP)
         GDENOM=GDTEMP+R(I,I)
            DO 16 IX=1,N
 16         GI(IX)=GNUM(IX)/GDENOM
C-----------------------------------------------------------
C        THIS IS THE FIRST GAIN COLUMN
C        CALCULATE THE COVARIANCE OF ERROR PI
C-----------------------------------------------------------
          DO 77 IP=1,N
             DO 79 JP=1,N
             PDUM(IP,JP)=(-1.*GI(IP))*HROW(JP)
            IF(IP.EQ.JP)PDUM(IP,JP)=1.+PDUM(IP,JP)
 79         CONTINUE
 77        CONTINUE
         CALL PRCD (PDUM,PKKM1,N,N,N,PI)
C------------------------------------------------------------
C        CALCULATE FIRST MEASUREMENT PREDICTION
C------------------------------------------------------------
C        CALL CZHAT(I,ZHAT)
         CALL CZHAT3(I,ZHAT,XB,YB,ZB)
C-----------------------------------------------------------------------
C        GFT WHITE NOISE,SCALE AND ADD TO TRUE MEASUREMENT TIME
C-----------------------------------------------------------------------
         CALL GGNML(DSEED,NR,RN)
         ZIC(I)=ZI(I)+RN(I)*.0001
C        ZIC(I)=ZI(I)
         ZDIFF(I)=ZIC(I)-ZHAT
C
     WRITE(6,90005)ZDIFF(I)
90005 FORMAT(1X,'ZDIFF',F15.8)
C----------------------------------------------------------------
C     COMPUTE THE GATE FOR ERRONEOUS TIME MEASUREMENTS
C----------------------------------------------------------------
C     PK1=DABS(PI(1,1))
      PK1= ABS(PI(1,1))
C     PK3=DABS(PI(3,3))
```

```
      PK3= ABS(PI(3,3))
C     PK5=DABS(PI(5,5))
      PK5= ABS(PI(5,5))
      IF((PK1.GE.PK3).AND.(PK1.GE.PK5))P=PK1
      IF((PK3.GE.PK1).AND.(PK3.GE.PK5))P=PK3
      IF((PK5.GE.PK1).AND.(PK5.GE.PK3))P=PK5
      PGATE=P/4860.
C     RGATE=DSQRT(DABS(R(I,I)))
      RGATE= SQRT( ABS(R(I,I)))
      GATE=3.*SQRT(PGATE+RGATE)
      WRITE(6,9000)GATE,I,ZDIFF(I)
 9000 FORMAT(1X,'GATE=',F12.7,4X,'ZDIFF(',I2,')=',F12.7)
C----------------------------------------------------|
C     EDIT INVALID TIME MEASUREMENTS                 |
C----------------------------------------------------|
      IF (ZDIFF(I).LT.GATE) GO TO 500
      WRITE(6,501)KK
 501  FORMAT('0','GATE HAS BEEN EXCEEDED TIME',I4)
      DO 502 LG=1,N
 502  GI(LG)=0.0
C----------------------------------------------------|
C      TAG INVALIC TIME MEASUREMENT                  |
C----------------------------------------------------|
      ZDIFF(I)=995.
C---------------------------------------------------------------------|
C     CALCULATE THE ESTIMATE BASED ON ONE MEASUREMENT                 |
C---------------------------------------------------------------------|
 500          DO 17 IZ=1,N
 17          XI(IZ)=XKKM1(IZ)+GI(IZ)*ZDIFF(I)
      WRITE(6,43)
 43   FORMAT('0',10X,'ZI',11X,'ZHAT',12X,'ZDIFF',13X,'GX',14X,'GXDT',15X
     1,'GY',16X,'GYDT',17X,'GZ','0')
      WRITE(6,44)ZI(I),ZHAT,ZDIFF,GI(1),GI(2),GI(3),GI(4),GI(5)
 44   FORMAT(' ',8F14.5)
      WRITE(6,148)
 148  FORMAT(' ',3X,'ZC',8X,'ZI',8X,'ZIC')
      WRITE(6,149)ZC(I),ZI(I),ZIC
 149  FORMAT(1X,3F14.5)
      WRITE(6,48)
 48   FORMAT(' ',3X,'H1',8X,'H2',8X,'H3',7X,'H4',8X,'H5')
      WRITE(6,49) HROW(1),HROW(2),HROW(3),HROW(4),HROW(5)
 49   FORMAT(1X,5F14.6)
      WRITE(6,35)
 35   FORMAT('0','XI AND XKKM1')
      DO 33 LC=1,N
 33   WRITE(6,34)XI(LC),XKKM1(LC)
 34   FORMAT(' ',2F11.4)
      WRITE(6,59)
```

```
59    FORMAT('0','PI MATRIX')
      DO 53 LK=1,N
53    WRITE(6,52) (PI(LK,LB), LB=1,N)
52    FORMAT(1X,5F14.4)
      WRITE(6,68)
68    FORMAT('0','PKKM1')
      DO 71 LE=1,N
71    WRITE(6,72)(PKKM1(LE,LF),LF=1,N)
72    FORMAT(1X,5F14.4)
C
          IF(I.EQ.4)GO TO 56
 12         DO 19  IQ=1,N
19            XKKM1(IQ)=XI(IQ)
               DO 23 IQ=1,N
                 DO 18 JQ=1,N
18                 PKKM1(IQ,JQ)=PI(IQ,JQ)
23               CONTINUE
97          CONTINUE
C-----------------------------------------------------------------------
C        NOTE-CALLED ORIGINAL X(0/-1),XKKM1. UPDATED AFTER 1
C        MEASUREMENT CALLED IT XI, THEN MADE XKKM1=XI AND WENT
C        THRU ITERATION AGAIN. AFTER YOU HAVE UPDATED XKKM1 FOR
C        EACH MEASUREMENT XKK=XI AND PKK=PI
C-----------------------------------------------------------------------
56        DO 57 ID=1,N
          XKK(ID)=XI(ID)
          XP6(ID,KK)=XI(ID)
           XKKM1(ID)=XI(ID)
            DO 58 JD=1,N
           PKK(ID,JD)=PI(ID,JD)
           P5(KK,ID,JD)=PI(ID,JD)
58        PKKM1(ID,JD)=PI(ID,JD)
57        CONTINUE
C
      XP7(KK)=(TRUX(KK)-XP6(1,KK))**2
      XP8(KK)=(TRUY(KK)-XP6(3,KK))**2
      XP9(KK)=(TRUZ(KK)-XP6(5,KK))**2
      IF(KK.NE.1) GO TO 666
      XP10(KK)=XP7(KK)
      XP11(KK)=XP8(KK)
      XP12(KK)=XP9(KK)
      GO TO 667
C
666   CONTINUE
         XP10(KK)=XP10(KK-1)+XP7(KK)
         XP11(KK)=XP11(KK-1)+XP8(KK)
         XP12(KK)=XP12(KK-1)+XP9(KK)
667   CONTINUE
```

```
      XP13(KK)=1/KK*XP10(KK)
      XP14(KK)=1/KK*XP11(KK)
      XP15(KK)=1/KK*XP12(KK)
C
      P1K(KK)=PI(1,1)
      P2K(KK)=PI(2,2)
      P3K(KK)=PI(3,3)
      P4K(KK)=PI(4,4)
      P5K(KK)=PI(5,5)
C
      WRITE(6,61)KK
   61 FORMAT(' ','TRUE POSITION TIME',I2)
      WRITE(6,62)TRUX(KK),TRUY(KK),TRUZ(KK)
   62 FORMAT(1X,3F11.4)
C
      WRITE(6,63)
   63 FORMAT('0','XKK')
      DO 37 LD=1,N
   37 WRITE(6,36) XKK(LD)
   36 FORMAT(1X,F11.4)
      WRITE(6,64)
   64 FORMAT('0','PKK')
      DO 65 LG=1,N
   65 WRITE(6,66)(PKK(LG,LH),LH=1,N)
   66 FORMAT(1X,5F14.4)
C-----------------------------------------------------------------------
C     RECALCULATE TIME MEASUREMENTS,FORM ABSOLUTE VALUE OF RESIDUALS
C-----------------------------------------------------------------------
      DO 81 I=1,4
C-----------------------------------------------------------------------
C     EDIT INVALID TIME MEASUREMENTS FOR ADAPTIVE MANEUVER ROUTINE
C-----------------------------------------------------------------------
      IF (ZDIFF(I).GE.999.) GO TO 82
C     CALL CZHAT(I,ZHAT)
      CALL CZHAT3(I,ZHAT,XB,YB,ZB)
C     ZDIFF(I)=DABS(ZIC(I)-ZHAT)
      ZDIFF(I)= ABS(ZIC(I)-ZHAT)
      GO TO 81
   82 ZDIFF(I)=0.0
      NZDIFF = NZDIFF- 1
   81 CONTINUE
C------------------------------------------------------------------
C     IF ALL TIME MEASUREMENTS EXCEED GATE BYPASS ADAPTIVE
C     MANEUVER ROUTINE
C------------------------------------------------------------------
      IF (NZDIFF.EQ.0) GO TO 80
      ZDIFAV=(ZDIFF(1)+ZDIFF(2)+ZDIFF(3)+ZDIFF(4))/NZDIFF
      WRITE(6,9900)
```

```
 9900 FORMAT(' ','ZDIFAV')
      WRITE(6,83)ZCIFAV
 83   FORMAT(5X,E14.5)
C-------------------------------------------------------------------
C     IF FILTER HAS NOT ACHIEVED STEADY STATE
C     DO NOT PERFORM ADAPTIVE MANEUVERING
C-------------------------------------------------------------------
      IF(KK.LE.4)GO TO 80
C-------------------------------------------------------------------
C     IF ZDIFAV MEETS CRITERIA TRANSFER OUT OF
C     ADAPTIVE MANEUVER ROUTINE
C-------------------------------------------------------------------
      IF(ZDIFAV.LE..10000D-04)GO TO 80
C-----------------------------
C     INCREASE THE GAIN
C-----------------------------
C     CALL QFIND(KK,SIGACC,SIGDIV,SIGCC,A14,Q)
      CALL ADD(PKK,Q,N,N,PKKM1)
C------------------------------------------------------------------------
C     PERFORM ADAPTIVE MANEUVERING BY REITERATING SAME TIME SLOT
C------------------------------------------------------------------------
C     WRITE(6,90007)
C0007 FORMAT(1X,'ADAPT')
      GO TO 711
 80   NZDIFF=4
      XDIFF1(KK)=XKK(1)-TRUX(KK)
      XDIFF3(KK)=XKK(3)-TRUY(KK)
      XDIFF5(KK)=XKK(5)-TRUZ(KK)
C------------------------------------------------------
C       CALCULATE THE PREDICTIONS FOR PKKM1
C------------------------------------------------------
      SI=0.
 1081 CONTINUE
C     CALL QFIND(KK,SIGACC,SIGDIV,SIGCC,A14,Q)

      DO 765 I=1,N
 765   WRITE(6,766)(Q(I,J),J=1,N)
 766   FORMAT(1X,6E18.5)
C
        CALL PROD(PHI,PKK,N,N,N,PHIPKK)
        CALL PROD(PHIPKK,PHIT,N,N,N,PKTEMP)
        CALL ADD(PKTEMP,Q,N,N,PKKM1)
C--------------------------------
C       CALCULATE NEW XKKM1
C--------------------------------
        CALL MMULT(PHI,XKK,N,N,XKKM1)
      IF(SI.EQ.0.)GO TO 1096
      WRITE(6,6669)
```

```
 6669 FORMAT('0','XI AND SMOOTHED XKKM1')
      DO 6670 LC=1,N
 6670 WRITE(6,6671) XI(LC),XKKM1(LC)
 6671 FORMAT(' ',2F11.4)
 1096 CONTINUE
         DO 41 IG=1,N
         SS(IG,KK)=XKKM1(IG)
 41      XP(IG,KK)=XKK(IG)
        DO 38 II=1,N
          DO 39 JJ=1,N
            SS1(KK,II,JJ)=PKKM1(II,JJ)
            SQ1(KK,II,JJ)=Q(II,JJ)
            P1(KK,II,JJ)=PKK(II,JJ)
  39      CONTINUE
 38     CONTINUE
C-----------------------------------------
C     CALCULATE SMOOTHED ESTIMATES
C-----------------------------------------
      IF(KK.EQ.1)GO TO 8888
      IF(SI.NE.0.)GO TO 8888
      CALL SMOOTH(SS,SS1,PHI,P5,XP6,KK,N)
C
      WRITE(6,9910)
 9910 FORMAT(' ','SMOOTHED P(K-1/K)')
      DO 6666 LE=1,N
         DO 6667 LK=1,N
            PKK(LE,LK)=P1(KK-1,LE,LK)
 6667    CONTINUE
 6666 CONTINUE
C
      DO 1095 LK=1,N
 1095 WRITE(6,6668)(PKK(LK,LE),LE=1,N)
 6668 FORMAT(1X,5F14.4)
      WRITE(6,1089)
 1089 FORMAT(' ','SMOOTHED XKK OF THE PREVIOUS TIME')
      DO 1084 I=1,N
          XKK(I)=XP(I,KK-1)
          WRITE(6,1087) XKK(I)
 1084 CONTINUE
 1087 FORMAT(1X,F11.4)
      XDIX(KK)=ABS(XKK(1)-TRUX(KK-1))
C
      IF(SI.EQ.1.)GO TO 1081
C
 8888 X9(KK)=KK
      P1P(KK)=PKK(1,1)
      P2P(KK)=PKK(2,2)
      P3P(KK)=PKK(3,3)
```

```
      P4P(KK)=PKK(4,4)
      P5P(KK)=PKK(5,5)
 99   CONTINUE
C----------------------------------------------------------------
C     THESE STATEMENTS ARE FOR ADDITIONAL PLOTTING
C GET DIOGONAL TERMS OF  P(K/N) AFTER  N-MEASUREMENTS.
C----------------------------------------------------------------
      DO 9992 LP=1,JTIME
         P1PP(LP)=P1(LP,1,1)
         P2PP(LP)=P1(LP,2,2)
         P3PP(LP)=P1(LP,3,3)
         P4PP(LP)=P1(LP,4,4)
         P5PP(LP)=P1(LP,5,5)
 9992 CONTINUE
C-----------------------------------
C  GET DIOGONAL TERMS OF P(K/K)
C-----------------------------------
      DO 9993 LP=1,ITIME
         P1P1(LP)=P5(LP,1,1)
         P2P2(LP)=P5(LP,2,2)
         P3P3(LP)=P5(LP,3,3)
         P4P4(LP)=P5(LP,4,4)
         P5P5(LP)=P5(LP,5,5)
 9993 CONTINUE
C----------------------------------------------------
C   GET SMOOTHED STATES AFTER N-MEASUREMENTS.
C----------------------------------------------------
      DO 9994 LP=1,JTIME
         XP1S(LP)=XP(1,LP)
         XP2S(LP)=XP(2,LP)
         XP3S(LP)=XP(3,LP)
         XP4S(LP)=XP(4,LP)
         XP5S(LP)=XP(5,LP)
 9994 CONTINUE
C-------------------------------------------------
C   GET ESTIMATION OF STATES FROM X(K/K)
C-------------------------------------------------
      DO 9995 LP=1,ITIME
         XP1K(LP)=XP6(1,LP)
         XP2K(LP)=XP6(2,LP)
         XP3K(LP)=XP6(3,LP)
         XP4K(LP)=XP6(4,LP)
         XP5K(LP)=XP6(5,LP)
 9995 CONTINUE
C--------------------------------------------------------------------
C   FIND ERROR BETWEEN TRUE POSITIONS AND KALMAN ESTIMATIONS.
C--------------------------------------------------------------------
      DO 9996 LP=1,ITIME
```

```
C         XKER(LP)=ABS(TRUX(LP)-XP1K(LP))
C         YKER(LP)=ABS(TRUY(LP)-XP3K(LP))
C         ZKER(LP)=ABS(TRUZ(LP)-XP5K(LP))
          XKER(LP)=TRUX(LP)-XP1K(LP)
          YKER(LP)=TRUY(LP)-XP3K(LP)
          ZKER(LP)=TRUZ(LP)-XP5K(LP)
 9996 CONTINUE
C-----------------------------------------------------------------
C   FIND ERROR BETWEEN TRUE POSITIONS AND SMOOTHED ESTIMATIONS.
C-----------------------------------------------------------------
      DO 9997 LP=1,JTIME
C         XSER(LP)=ABS(TRUX(LP)-XP1S(LP))
C         YSER(LP)=ABS(TRUY(LP)-XP3S(LP))
C         ZSER(LP)=ABS(TRUZ(LP)-XP5S(LP))
          XSER(LP)=TRUX(LP)-XP1S(LP)
          YSER(LP)=TRUY(LP)-XP3S(LP)
          ZSER(LP)=TRUZ(LP)-XP5S(LP)
 9997 CONTINUE
C---------------------------------------
C   GET COVARIANCE ERROR ELLIPSOIDS
C---------------------------------------
      LK=1
      DO 9998 LP=18,ITIME,18
      ANGL(LK)=0.5*ATAN(2.*P6P6(LP)/(P1P1(LP)-P3P3(LP)))
      ANGLD(LK)=ANGL(LK)*180./3.14159
      SIG1(LK)=(P1P1(LP)+P3P3(LP))/2.
      SIG2(LK)=P6P6(LP)/SIN(2.*ANGL(LK))
      SIGX(LK)=SIG1(LK)+SIG2(LK)
      SIGY(LK)=SIG1(LK)-SIG2(LK)
      SX(LK)=(SIGX(LK)**.5)*25.
      SY(LK)=(SIGY(LK)**.5)*25.
      LK=LK+1
 9998 CONTINUE
      WRITE(6,99960)
99960 FORMAT(' ',' ANGLES IN RADIAN')
      WRITE(6,*)ANGL
      WRITE(6,99963)
99963 FORMAT(' ',' ANGLES IN DEGREE')
      WRITE(6,*)ANGLD
      WRITE(6,99961)
99961 FORMAT(' ',' SIGMA X')
      WRITE(6,*)SX
      WRITE(6,99962)
99962 FORMAT(' ',' SIGMA Y')
      WRITE(6,*)SY
C
C
      PT=3.14159265/12.
```

```
      K=18
      DO 44445 J=1,7
      CT=COS(ANGL(J))
      ST=SIN(ANGL(J))
      DO 44446 I=1,25
      XII=I
      XP88(J,I)=SX(J)*COS(PT*XII)*CT-SY(J)*SIN(PT*XII)*ST+TRUX(K)
      YP88(J,I)=SX(J)*COS(PT*XII)*ST+SY(J)*SIN(PT*XII)*CT
44446 CONTINUE
      K=K+18
44445 CONTINUE
      DO 44447 LP=1,25
      XP19(LP)=XP88(1,LP)
      XP29(LP)=XP88(2,LP)
      XP39(LP)=XP88(3,LP)
      XP49(LP)=XP88(4,LP)
      XP59(LP)=XP88(5,LP)
      XP69(LP)=XP88(6,LP)
      XP79(LP)=XP88(7,LP)
      YP19(LP)=YP88(1,LP)
      YP29(LP)=YP88(2,LP)
      YP39(LP)=YP88(3,LP)
      YP49(LP)=YP88(4,LP)
      YP59(LP)=YP88(5,LP)
      YP69(LP)=YP88(6,LP)
      YP79(LP)=YP88(7,LP)
44447 CONTINUE
C
      WRITE(6,44448)
44448 FORMAT(' ','X OF ELIPSE')
      WRITE(6,*)XP29
      WRITE(6,44449)
44449 FORMAT(' ','Y OF ELIPSE')
      WRITE(6,*)YP29
      WRITE(6,44450)XP19,YP19,XP29,YP29,XP39,YP39,XP49,YP49,XP59,YP59,
     1XP69,YP69,XP79,YP79
44450 FORMAT(8F9.2)
C
C     WRITE(6,9916)
C9916 FORMAT('1')
C     CALL PLOTP(TRUX,TRUY,201,4)
C     CALL PLOTP(X9,XDIFF1,31,1)
C     CALL PLOTP(X9,XDIX,30,3)
C     WRITE(6,90007)
C0007 FORMAT('1')
C     CALL PLOTP(X9,P1P,30,1)
C     CALL PLOTP(X9,P1K,31,3)
C     WRITE(6,90008)
```

```
C0008 FORMAT('1')
C       CALL  PLOTP(X9,P2P,30,1)
C       CALL  PLOTP(X9,P2K,31,3)
C       WRITE(6,90009)
C0009 FORMAT('1')
C       CALL  PLOTP(X9,P3P,30,1)
C       CALL  FLOTP(X9,P3K,31,3)
C       WRITE(6,90010)
C0010 FORMAT('1')
C       CALL  PLOTP(X9,P4P,30,1)
C       CALL  PLOTP(X9,P4K,31,3)
C       WRITE(6,90011)
C0011 FORMAT('1')
C       CALL  PLOTP(X9,P5P,30,1)
C       CALL  FLOTP(X9,P5P,31,3)
C       WRITE(6,90012)
C0012 FORMAT('1')
C       CALL  PLOTP(X9,XSER,30,1)
C       CALL  PLOTP(X9,XKER,31,3)
C       WRITE(6,90013)
C0013 FORMAT('1')
C       CALL  PLOTP(X9,P1PP,30,1)
C       CALL  PLOTP(X9,P1P1,31,3)
C       WRITE(6,90014)
C0014 FORMAT('1')
C       CALL  PLOTP(X9,P2PP,30,1)
C       CALL  PLOTP(X9,P2P2,31,3)
C       WRITE(6,90015)
C0015 FORMAT('1')
C       CALL  PLOTP(X9,P3PP,30,1)
C       CALL  PLOTP(X9,P3P3,31,3)
C       WRITE(6,90016)
C0016 FORMAT('1')
C       CALL  FLOTP(X9,P4PP,30,1)
C       CALL  PLOTP(X9,P4P4,31,3)
C       WRITE(6,90017)
C0017 FORMAT('1')
C       CALL  PLOTP(X9,P5PP,30,1)
C       CALL  PLOTP(X9,P5P5,31,3)
C
        WRITE(6,90018)
90018 FORMAT('1')
        WRITE(6,90019)
90019 FORMAT(' ','ERROR OF KALMAN')
        WRITE(6,*)XKER
        WRITE(6,90020)
90020 FORMAT(' ','ERROR AFTER SMOOTHING')
        WRITE(6,*)XSER
```

```
      WRITE(6,90024)
90024 FORMAT(' ',' TRUE X-POSITION')
      WRITE(6,*)TRUX
      WRITE(6,90025)
90025 FORMAT(' ','TRUE Y-POSITION')
      WRITE(6,*)TRUY
      WRITE(6,90026)
90026 FORMAT(' ','TRUE Z-POSITION')
      WRITE(6,*)TRUZ
      WRITE(6,90021)
90021 FORMAT(' ','ESTIMATES OF KALMAN')
      WRITE(6,*)XP1K
      WRITE(6,*)XP2K
      WRITE(6,*)XP3K
      WRITE(6,*)XP4K
      WRITE(6,*)XP5K
      WRITE(6,90022)
90022 FORMAT(' ','ESTIMATES AFTER SMOOTHING')
      WRITE(6,*)XP1S
      WRITE(6,*)XP2S
      WRITE(6,*)XP3S
      WRITE(6,*)XP4S
      WRITE(6,*)XP5S
      WRITE(6,90023)
90023 FORMAT(' ','DIOGONOL TERMS OF COVARIANCE')
      WRITE(6,*)P1P1
      WRITE(6,*)P2P2
      WRITE(6,*)P3P3
      WRITE(6,*)P4P4
      WRITE(6,*)P5P5
      WRITE(6,*)P1PP
      WRITE(6,*)P2PP
      WRITE(6,*)P3PP
      WRITE(6,*)P4PP
      WRITE(6,*)P5PP
      WRITE(6,99950)
99950 FORMAT('1',5X,'TIME',7X,'TRUE-X',7X,'X-AFTER KALMAN',5X,'X-AFTER
     *SMOOTHING',/,5X,'-----------------------------------------------------
     *---------------')
      DO 99951 IP=1,JTIME
      WRITE(6,99952)IP,TRUX(IP),XP1K(IP),XP1S(IP)
99952 FORMAT('0',5X,I3,6X,F11.4,6X,F11.4,6X,F11.4)
99951 CONTINUE
      WRITE(6,99953)
99953 FORMAT('1',5X,'TIME',7X,'TRUE-Y',7X,'Y-AFTER KALMAN',5X,'Y-AFTER
     *SMOOTHING',/,5X,'-----------------------------------------------------
     *---------------')
      DO 99954 IP=1,JTIME
```

```
      WRITE(6,99955)IP,TRUY(IP),XP3K(IP),XP3S(IP)
99955 FORMAT('0',5X,I3,6X,F11.4,6X,F11.4,6X,F11.4)
99954 CONTINUE
      WRITE(6,99958)
99958 FORMAT('1',5X,'TIME',7X,'TRUE-Z',7X,'Z-AFTER KALMAN',5X,'Z-AFTER
     *SMOOTHING',/,5X,'--------------------------------------------------
     *-------------')
      DO 99956 IP=1,JTIME
      WRITE(6,99957)IP,TRUZ(IP),XP5K(IP),XP5S(IP)
99957 FORMAT('0',5X,I3,6X,F11.4,6X,F11.4,6X,F11.4)
99956 CONTINUE
      CALL PLOTG(TRUX,TRUY,ITIME,1,1,1,'X-POSITION(FT)',14,
     *'Y-POSITION(FT)',14,2000.,8000.,1000.,1500.,3.5,5.5)
      CALL PLOTG(X9,XSER,JTIME,1,1,1,'TIME SLOTS',10,'ERROR IN X-POS.(FT
     *)',19,1.,50.,-10.,10.,7.,4.)
      CALL PLOTG(X9,XKER,ITIME,2,1,2,'TIME SLOTS',10,'ERROR IN X-POS.(FT
     *)',19,1.,50.,-10.,10.,7.,4.)
      CALL PLOTG(X9,YSER,JTIME,1,1,1,'TIME SLOTS',10,'ERROR IN Y-POS.(FT
     *)',19,1.,50.,-10.,10.,7.,4.)
      CALL PLOTG(X9,YKER,ITIME,2,1,2,'TIME SLOTS',10,'ERROR IN Y-POS.(FT
     *)',19,1.,50.,-10.,10.,7.,4.)
      CALL PLOTG(X9,ZSER,JTIME,1,1,1,'TIME SLOTS',10,'ERROR IN Z-POS.(FT
     *)',19,1.,50.,-10.,10.,7.,4.)
      CALL PLOTG(X9,ZKER,JTIME,2,1,2,'TIME SLOTS',10,'ERROR IN Z-POS.(FT
     *)',19,1.,50.,-10.,10.,7.,4.)
      CALL PLOTG(X9,P1PP,JTIME,1,1,1,'TIME SLOTS',10,'MEAN SQUARE ESTIMA
     *TION ERROR OF X',33,1.,50.,0.,5000.,7.,4.)
      CALL PLOTG(X9,P1P1,ITIME,2,1,2,'TIME SLOTS',10,'MEAN SQUARE ESTIMA
     *TION ERROR OF X',33,1.,50.,0.,5000.,7.,4.)
      CALL PLOTG(X9,P2PP,JTIME,1,1,1,'TIME SLOTS',10,'MEAN SQUARE ESTIMA
     *TION ERROR OF X-VEL',37,1.,50.,0.,30.,7.,4.)
      CALL PLOTG(X9,P2P2,ITIME,2,1,2,'TIME SLOTS',10,'MEAN SQUARE ESTIMA
     *TION ERROR OF X-VEL',37,1.,50.,0.,30.,7.,4.)
      CALL PLOTG(X9,P3PP,JTIME,1,1,1,'TIME SLOTS',10,'MEAN SQUARE ESTIMA
     *TION ERROR OF Y',33,1.,50.,0.,300.,7.,4.)
      CALL PLOTG(X9,P3P3,ITIME,2,1,2,'TIME SLOTS',10,'MEAN SQUARE ESTIMA
     *TION ERROR OF Y',33,1.,50.,0.,300.,7.,4.)
      CALL PLOTG(X9,P4PP,JTIME,1,1,1,'TIME SLOTS',10,'MEAN SQUARE ESTIMA
     *TION ERROR OF Y-VEL',37,1.,50.,0.,20.,7.,4.)
      CALL PLOTG(X9,P4P4,ITIME,2,1,2,'TIME SLOTS',10,'MEAN SQUARE ESTIMA
     *TION ERROR OF Y-VEL',37,1.,50.,0.,20.,7.,4.)
      CALL PLOTG(X9,P5PP,JTIME,1,1,1,'TIME SLOTS',10,'MEAN SQUARE ESTIMA
     *TION ERROR OF Z',33,1.,50.,150.,600.,7.,4.)
      CALL PLOTG(X9,P5P5,ITIME,2,1,2,'TIME SLOTS',10,'MEAN SQUARE ESTIMA
     *TION ERROR OF Z',33,1.,50.,150.,600.,7.,4.)
      CALL PLOT(0.0,0.0,999)
      STOP
      END
```

```
      SUBROUTINE TRAJEC(K,DATR,ZI,TD)
C
C-----------------------------------------------------------------------
C       SUBROUTINE COMPUTES TRUE TRAJECTORY OF TORPEDO
C
C   DATR(1)=TRUE X POSITION,  DATR(2)=TRUE Y POSITION
C   DATR(3)=TRUE Z POSITION,  DATR(4)=TRUE X VELOCITY
C   DATR(5)=TRUE Y VELOCITY,  DATR(6)=TRUE Z VELOCITY,  DATR(7)=0.0,
C   DATR(8)=0.0,  DATR(9)=0.0,  DATR(10)=0.0,  DATR(11)=0.0,
C   DATR(12)=0.0,  DATR(13)=ACCELLERATION ANGLE(RADIANS),
C   DATR(14)=TURN RATE(RAD/SEC,  DATR(15)=SAMPLE TIME
C   DATR(16)=HORIZONTAL ACCELLERATION(FT/SEC2)
C   DATR(17),  DATR(18)TIME SLOT INTERVAL DURING WHICH MANEUVER
C       IS PERFORMEC,ZI=TRUE TIMES,TD=TRUE POSITIONS
C-----------------------------------------------------------------------
C     DOUBLE PRECISION DATR,ZI
      REAL*4    DATR(8),ZI(4),TD(3)
      T=0.0
      VEL=4860.
C     RANGF=DSQRT(DATR(1)*DATR(1)+DATR(2)*DATP(2)+DATR(3)*DATR(3))
      RANGE= SQRT(DATR(1)*DATR(1)+DATR(2)*DATR(2)+DATR(3)*DATR(3))
      I=1
      ZI(I)=1./VEL*(((CATR(1)+15.)**2)+((DATR(2)+15.)**2)
     1       +((DATR(3)-15.)**2))**0.5
      I=2
      ZI(I)=1./VEL*(((CATR(1)-15.)**2)+((DATR(2)+15.)**2)
     1       +((DATR(3)-15.)**2))**0.5
      I=3
      ZI(I)=1./VEL*(((DATR(1)+15.)**2)+((DATR(2)-15.)**2)
     1       +((DATR(3)-15.)**2))**0.5
      I=4
      ZI(I)=1./VEL*(((DATR(1)+15.)**2)+((DATR(2)+15.)**2)
     1       -((CATR(3)-15.)**2))**0.5
      DO 9 I=1,3
      TD(I)=DATR(I)
 9    CONTINUE
      IF((K.LE.DATR(17)).AND.(K.GT. DATR(16))) GO TO 11
      DATR(7)=0.0
      DATR(8)=0.0
      DATR(14)=1.31
      GO TO 12
 11   DATR(14)=.005
 13   DATR(12)=DATR(12)+DATR(13)*DATR(14)
C     DATR(7)=DATR(16)*DCOS(DATR(13))
      DATR(7)=DATR(15)*COS(DATR(12))
C     DATR(8)=DATR(16)*DSIN(DATR(13))
      DATR(8)=DATR(15)*SIN(DATR(12))
 12   DO 10 I=1,5
```

```
      DATR(I)=DATR(I)+DATR(I+3)*DATR(14)+(((DATR(14))**2)/2)*DATR(I+6)
10    CONTINUE
      T=T+DATR(14)
      IF(ABS(T-1.31).LE.0.0001) RETURN
      GO TO 13
      END
```

```
      SUBROUTINE TRAJC3(DATR,ZI,TD,XB,YB,ZB)
C
C-----------------------------------------------------------
C THIS SUBROUTINE IS USED DURING MULTIPLE ARRAY TRACKING
C-----------------------------------------------------------
      REAL*4 DATR(17),ZI(4),TD(3),XB(4),YB(4),ZB(4)
      T=0.0
      VEL=4860.
      RANGE=SQRT(DATR(1)*DATR(1)+DATR(2)*DATR(2)+DATR(3)*DATR(3))
      DO 5 I=1,4
         ZI(I)=1./VEL*(((DATR(1)-XB(I))**2)+((DATR(2)-YB(I))**2)
     1     +((DATR(3)-ZB(I))**2))**0.5
    5 CONTINUE
      DO 9 I=1,3
      TD(I)=DATR(I)
 9    CONTINUE
      IF((K.LE.DATR(17)).AND.(K.GT. DATR(16))) GO TO 11
      DATR(7)=0.0
      DATR(8)=0.0
      DATR(14)=1.31
      GO TO 12
 11   DATR(14)=.005
 13   DATR(12)=DATR(12)+DATR(13)*DATR(14)
C     DATR(7)=DATR(16)*DCOS(DATR(13))
      DATR(7)=DATR(15)*COS(DATR(12))
C     DATR(8)=DATR(16)*DSIN(DATR(13))
      DATR(8)=DATR(15)*SIN(DATR(12))
 12   DO 10 I=1,5
      DATR(I)=DATR(I)+DATR(I+3)*DATR(14)+(((DATR(14))**2)/2)*DATR(I+6)
 10   CONTINUE
      T=T+DATR(14)
      IF(ABS(T-1.31).LE.0.0001) RETURN
      GO TO 13
      END
```

```
      SUBROUTINE INIT (XKKM1,PKKM1)
C
C--------------------------------------------------------------------
C THIS ROUTINE IS TO INITIALIZE THE ARRAYS XKKM1 AND PKKM1.
C THESE VARIABLES ARE PART OF A COMMON BLOCK.
C--------------------------------------------------------------------
C
      DOUBLE PRECISION XKKM1,PKKM1
      REAL*4    XKKM1(5),PKKM1(5,5)
      DO 20 J=1,5
         DO 10 I=1,5
            PKKM1(I,J)=0.0
   10 CONTINUE
   20 CONTINUE
      XKKM1(1)=7475.
      XKKM1(2)=-40.
      XKKM1(3)=1275.
      XKKM1(4)=0.0
      XKKM1(5)=0.
      DO 30 I=1,5
         PKKM1(I,I)=1000.
   30 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CZHAT(I,ZHAT)
C
C-----------------------------------------------------------------------
C   THIS SUBROUTINE COMPUTES ESTIMATES OF TRANSIT TIME MEASUREMENTS
C   FOR SINGLE ARRAY TRACKING.
C-----------------------------------------------------------------------
C
      DOUBLE PRECISION XKKM1,ZHAT
      COMMON/THE1/XKKM1(5)
      REAL*4 ZHAT
      VEL=4860.
      IF(I.EQ.1)ZHAT=1./VEL*(((XKKM1(1)+15.)**2)+((XKKM1(3)+15.)**2)
     1+((XKKM1(5)+15.)**2))**.5
      IF(I.EQ.2)ZHAT=1./VEL*(((XKKM1(1)-15.)**2)+((XKKM1(3)+15.)**2)
     1+((XKKM1(5)+15.)**2))**.5
      IF(I.EQ.1)ZHAT=1./VEL*(((XKKM1(1)+15.)**2)+((XKKM1(3)-15.)**2)
     1+((XKKM1(5)+15.)**2))**.5
      IF(I.EQ.1)ZHAT=1./VEL*(((XKKM1(1)+15.)**2)+((XKKM1(3)+15.)**2)
     1+((XKKM1(5)-15.)**2))**.5
      RETURN
      END
```

```
      SUBROUTINE CZHAT3 (I,ZHAT,XB,YB,ZB)
C
C-----------------------------------------------------------------------
C     THIS SUBROUTINE COMPUTES ESTIMATES OF TRANSIT TIME MEASUREMENTS
C     FOR MULTIPLE ARRAY TRACKING
C-----------------------------------------------------------------------
C
      DOUBLE PRECISION XKKM1,ZHAT
      REAL*4     XB(4),YB(4),ZB(4)
C     REAL*4           PKKM1(5,5),PKK(5,5),XKK(5)
      COMMON/THE1/XKKM1(5)
      VEL=4860.
      XO=XB(I)
      YO=YB(I)
      ZO=ZB(I)
      ZHAT=(1./VEL)*(((XKKM1(1)-XO)**2)+((XKKM1(3)-YO)**2)+((XKKM1(5)
     1-ZO)**2))**0.5
      RETURN
      END
```

```
      SUBROUTINE CHROW3 (I,HROW,XB,YB,ZB)
C
C---------------------------------------------------------------
C     THIS SUBROUTINE USED FOR MULTIPLE ARRAY TRACKING
C---------------------------------------------------------------
C
      DOUBLE PRECISION XKKM1,HROW,DENOM
      REAL*4     HROW(5),XB(4),YB(4),ZB(4)
      COMMON/THE1/XKKM1(5)
      VEL=4860.
      XO=XB(I)
      YO=YB(I)
      ZO=ZB(I)
      DENOM=(((XKKM1(1)-XO)**2)+((XKKM1(3)-YO)**2)+((XKKM1(5)-ZO)
     1**2))**0.5
      HROW(1)=(1./VEL)*((XKKM1(1)-XO)/DENOM)
      HROW(3)=(1./VEL)*((XKKM1(3)-YO)/DENOM)
      HROW(5)=(1./VEL)*((XKKM1(5)-ZO)/DENOM)
      DO 27 J=2,4,2
 27    HROW(J)=0.
      RETURN
      END
```

```
      SUBROUTINE CHROW(I,HROW)
C
C-----------------------------------------------------------------
C    THIS SUBROUTINE USED FOR SINGLE ARRAY TRACKING
C-----------------------------------------------------------------
C     DOUBLE PRECISION XKKM1,DENOM1,DENOM2,DENOM3,DENOM4,DENOM,HROW,
C    1H1,H3,H5
      COMMON/THE1/XKKM1(5)
      REAL*4 HROW(5)
      VEL=4860.
      DENOM1=(((XKKM1(1)+15.)**2)+((XKKM1(3)+15.)**2)+((XKKM1(5)+15.)**
     12))**0.5
      DENOM2=(((XKKM1(1)-15.)**2)+((XKKM1(3)+15.)**2)+((XKKM1(5)+15.)**
     12))**0.5
      DENOM3=(((XKKM1(1)+15.)**2)+((XKKM1(3)-15.)**2)+((XKKM1(5)+15.)**
     12))**0.5
      DENOM4=(((XKKM1(1)+15.)**2)+((XKKM1(3)+15.)**2)+((XKKM1(5)-15.)**
     12))**0.5
      A1=1.
      A2=1.
      A3=1.
      DENOM=DENOM1
      IF(I.EQ.2)DENOM=DENOM2
      IF(I.EQ.3)DENOM=DENOM3
      IF(I.EQ.4)DENOM=DENOM4
      IF(I.EQ.2)A1=-1.
      H1=(1./VEL)*((XKKM1(1)+A1*15.)/DENOM)
      IF(I.EQ.3)A2=-1.
      H3=(1./VEL)*((XKKM1(3)+A2*15.)/DENOM)
      IF(I.EQ.4)A3=-1.
      H5=(1./VEL)*((XKKM1(5)+A3*15.)/DENCM)
      HROW(1)=H1
      HROW(3)=H3
      HROW(5)=H5
      DO 27 J=2,4,2
   27   HROW(J)=0.
      RETURN
      END
```

```
      SUBROUTINE QFIND(K,SIGAAC,SIGDIV,SIGCC,A,Q)
C
C-------------------------------------------------------------
C     THIS SUBROUTINE COMPUTES THE ADAPTIVE Q MATRIX
C-------------------------------------------------------------
C     DOUBLE PRECISION XHKKM1,PKKM1,PKK,XHKK,Q
      REAL*4    Q(5,5)
      COMMON/THE1/XKKM1(5)/THE2/PKKM1(5,5),PKK(5,5),XKK(5)
      IF(K.NE.1) GO TO 15
      DO 10 I=1,5
         DO 10 J=1,5
   10 Q(I,J)=0.0
      SIGAAC=SIGAAC**2
      Q(5,5)=(SIGDIV*A)**2
      SIGCC=SIGCC**2
      G1=(A**2)/2.0
      G2=G1**2
      G3=A*G1
      G4=A**2
 15   A1=XKK (2)**2+XKK (4)**2
      A3=XKK (2)/SQRT(A1)
      B=XKK (2)
      C=XKK (4)/SQRT(A1)
      D=XKK (2)
C     D1=D**2
      E1=(A3**2)*SIGACC+(B**2)*SIGCC
      E12=A3*C*SIGACC-B*D*SIGCC
      E2=(C**2)*SIGACC+(D**2)*SIGCC
C     E13=((B*D)/A2)*SIGHAC
C     E23=((C*D)/A2)*SIGHAC
C     E3=A1*SIGVCC+((D**2)*A1*SIGHAC )/(A2**2)+((D1**2)*SIGVAC)/(A2**2)
      Q(1,1)=G2*E1
      Q(1,2)=G3*E1
      Q(1,3)=G2*E12
      Q(1,4)=G3*E12
      Q(2,2)=A2*E1
      Q(2,3)=G3*E12
      Q(2,4)=A2*E12
      Q(3,3)=G2*E2
      Q(3,4)=G3*E2
      Q(4,4)=A2*E2
      DO 27 I=1,4
         DO 27 J=1,I
   27 Q(I,J)=Q(J,I)
      RETURN
      END
```

```
      SUBROUTINE SMOOTH(SS,SS1,PHI,P5,XP6,KK,N)
C
C-----------------------------------------------------------------
C    THIS SUBROUTINE COMPUTES SMOOTHED STATES AND COVARIANCES
C-----------------------------------------------------------------
      COMMON/THE3/XP(5,210)/THE4/P1(210,5,5)/THE5/SI
C     DOUBLE PRECISION XP,SS,P1,SS1,SQ1,XKKS,PKKS
      REAL*4      XNNM1(5),PNNM1(5,5),SS(5,210),SS1(210,5,5),
     1            XP1(5),SS2(5),P2(5,5),SS3(5,5),P5(210,5,5),XP6(5,210),
     2            PHI(5,5),PHIT(5,5),SS3R(5,5),AK(5,5),AKT(5,5),
     3            PKKS(5,5),XKKS(5),TEMP1(5,5),TEMP2(5),TEMP3(5),
     4            TEMP4(5,5),TEMP5(5,5),TEMP6(5,5),CH(5,5)
      L=KK-1
      DO 20 K=1,L
      K1=L-K+1
      DO 21 I=1,N
         XP1(I)=XP6(I,K1)
         SS2(I)=SS(I,K1)
   21 CONTINUE
C
      DO 22 I=1,N
         DO 23 J=1,N
              P2(I,J)=P5(K1,I,J)
              SS3(I,J)=SS1(K1,I,J)
   23    CONTINUE
   22 CONTINUE
C-----------------------------------
C     FIND TRANSPCSE OF PHI
C-----------------------------------
      CALL TRANS(PHI,N,N,PHIT)
C-----------------------------------
C     FIND INVERSE OF P(K+1/K)
C-----------------------------------
      CALL RECIP(SS3,N,SS3R)
      CALL PROD(SS3,SS3R,N,N,N,CH)
C-----------------------------------------------------------------
C     CALCULATE A(K)=P(K/K)*TRANSPOSE(Q(K))*INVERSE(P(K+1/K))
C-----------------------------------------------------------------
      CALL PROD(PHIT,SS3R,N,N,N,TEMP1)
      CALL PROD(P2,TEMP1,N,N,N,AK)
      DO 27 I=1,N
   27    XNNM1(I)=XP(I,K1+1)
C---------------------------------------------------
C      CALCULATE SMOOTHED ESTIMATES OF STATES
C---------------------------------------------------
      CALL SUB(XNNM1,SS2,N,1,TEMP2)
      CALL PROD(AK,TEMP2,N,N,1,TEMP3)
      CALL ADD(XP1,TEMP3,N,1,XKKS)
```

```
C
      DO 80 I=1,N
   80    XP(I,K1)=XKKS(I)
C
      DO 31 I=1,N
         DO 32 J=1,N
   32          PNNM1(I,J)=P1(K1+1,I,J)
   31 CONTINUE
C-------------------------------------------
C     FIND SMOOTHEC COVARIANCES
C-------------------------------------------
      CALL SUB(PNNM1,SS3,N,N,TEMP4)
      CALL TRANS(AK,N,N,AKT)
      CALL PROD(TEMP4,AKT,N,N,N,TEMP5)
      CALL PROD(AK,TEMP5,N,N,N,TEMP6)
      CALL ADD(P2,TEMP6,N,N,PKKS)
C
      DO 38 I=1,N
         DO 39 J=1,N
   39         P1(K1,I,J)=PKKS(I,J)
   38 CONTINUE
C
   20 CONTINUE
      IS=1.
      RETURN
      END
```

```
      SUBROUTINE RECIP(A,N,C)
C
C     DOUBLE PRECISION A,D,C
      DIMENSION A(5,5),D(5,10),C(5,5)
C
      DO 10 K=1,N
      DO 11 J=1,N
   11 D(K,J)=A(K,J)
   10 CONTINUE
      DO 12 K=1,N
      I=K+N
      DO 14 J=6,10
      IF(I.NE.J)GO TO 13
      D(K,J)=1.
      GO TO 14
   13 D(K,J)=0.
   14 CONTINUE
   12 CONTINUE
      DO 15 K=1,N
      M=K+1
      DO 16 J=M,10
   16 D(K,J)=D(K,J)/D(K,K)
      D(K,K)=1.
      DO 17 L=1,N
      IF(L.EQ.K)GC TO 17
      DO 19 I=1,10
      IF(I.EQ.K)GC TO 19
      D(L,I)=D(L,I)-D(L,K)*D(K,I)
   19 CONTINUE
      D(L,K)=0.
   17 CONTINUE
   15 CONTINUE
      DO 20 K=1,N
      DO 21 J=1,N
      I=J+N
   21 C(K,J)=D(K,I)
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PROD(A,B,N,M,L,C)
C
C     DOUBLE PRECISION A,B,C
      REAL*4    A(N,M),B(M,L),C(N,L)
      DO 1 I=1,N
      DO 1 J=1,L
1     C(I,J)=0.
      DO 2 I=1,N
      DO 2 J=1,L
      DO 2 K=1,M
2     C(I,J)=C(I,J)+A(I,K)*B(K,J)
      RETURN
      END
```

```
      SUBROUTINE MMULT(A,B,N,M,C)
C
C     DOUBLE PRECISION A,B,C
      REAL*4     A(N,M),B(M),C(N)
      DO 3 I=1,N
      C(I)=0.
      DO 4 J=1,M
4     C(I)=C(I)+A(I,J)*B(J)
3     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE VMULT(A,B,N,C)
C
C     DOUBLE PRECISION A,B
      REAL*4     A(N),B(N)
      C=0.
      DO 6 I=1,N
6     C=C+A(I)*B(I)
      RETURN
      END
```

```
      SUBROUTINE TRANS(A,N,M,B)
C
C     DOUBLE PRECISION A,B
      REAL*4     A(N,M),B(M,N)
      DO 13 I=1,N
      DO 13J=1,M
13    B(J,I)=A(I,J)
      RETURN
      END
```

```
      SUBROUTINE ADD (A,B,N,M,C)
C
C     DOUBLE PRECISION A,B,C
      REAL*4     A(N,M),B(N,M),C(N,M)
      DO 15 I=1,N
      DO 15 J=1,M
15    C(I,J)=A(I,J)+B(I,J)
      RETURN
      END
```

```
      SUBROUTINE SUB(A,B,N,M,C)
C
C     DOUBLE PRECISION A,B,C
      DIMENSION A(N,M),B(N,M),C(N,M)
      DO 10 I=1,N
         DO 10 J=1,M
   10          C(I,J)=A(I,J)-B(I,J)
      RETURN
      END
```

LIST OF REFERENCES

1. Gelb, A., Applied Optimal Estimation, M.I.T. Press, 1974.

2. Technical Manual, NAVORD OD 41964, NAVTOPRSTA Keyport Range Complex and Associated Data, May 1970.

3. O'Brien, Paul A., An Application of Kalman Filtering to Torpedo Tracking, Master's Thesis, Naval Postgraduate School, September 1980.

4. Dwyer, Dennis M., Real Time Kalman Filtering for Torpedo Range Tracking, Master's Thesis, Naval Postgraduate School, December 1978.

5. Sorenson, H. W., Advances in Control Systems, Volume 3, 1966.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 62 Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
4.	Professor H. A. Titus, Code 62Ts Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	2
5.	Professor Alex Gerba, Code 62Gz Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
6.	Deniz Kuvvetleri Komutanligi Egitim Dairesi Ankara/Turkey	5
7.	Istanbul Teknik Universitesi Elektrik Fakültesi Gümüssuyu, Istanbul/Turkey	1
8.	Bogazici Universitesi P.K. 2 Bebek, Istanbul/Turkey	1
9.	Orta Dogu Teknik Universitesi Ankara/Turkey	1
10.	Mustafa Işık Dogancilar, Ramazanoglu sok No: 10/5 Üsküdar, Istanbul/Turkey	2

Thesis 198783
I725 Isik
c.1
An application of Kalman filtering and smoothing to torpedo tracking.